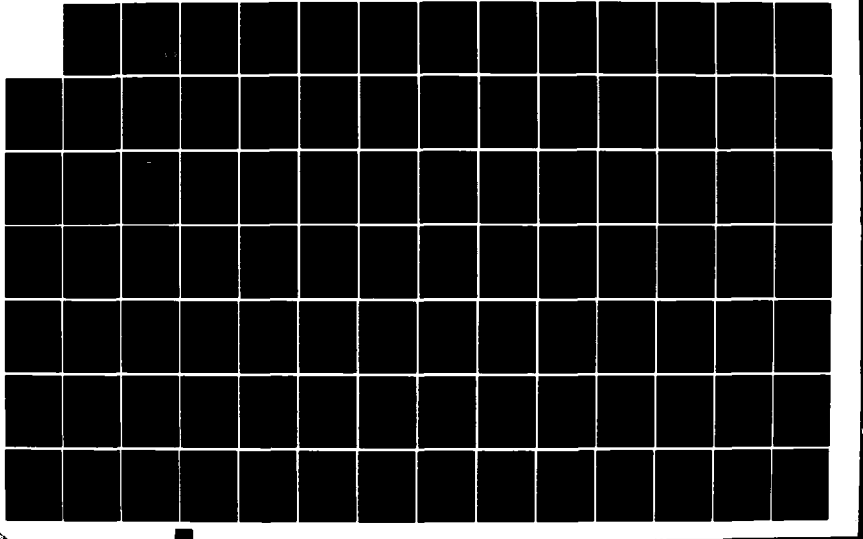
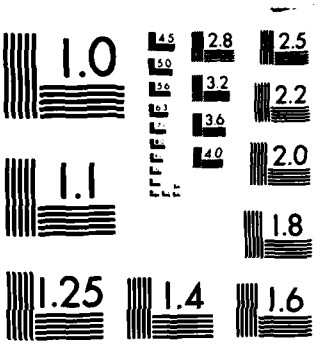
AD-A151 847 AN ANALYSIS OF GRID-BASED LINE DRAWING QUANTIZATION METHODS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING L A VALLADO DEC 84 1/2

UNCLASSIFIED AFIT/GCS/ENG/84D-31 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963 A

AN ANALYSIS OF GRID-BASED LINE DRAWING QUANTIZATION METHODS

THESIS

Laura A. Vallado
Captain, USAF

AFIT/GCS/ENG/84D-31

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

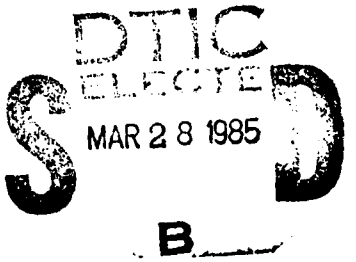
AFIT/GCS/ENG/84D-31

AN ANALYSIS OF GRID-BASED LINE DRAWING QUANTIZATION METHODS

THESIS

Laura A. Vallado
Captain, USAF

AFIT/GCS/ENG/84D-31



AFIT/GCS/ENG/84D-31

AN ANALYSIS OF GRID-BASED LINE DRAWING QUANTIZATION METHODS

THESIS

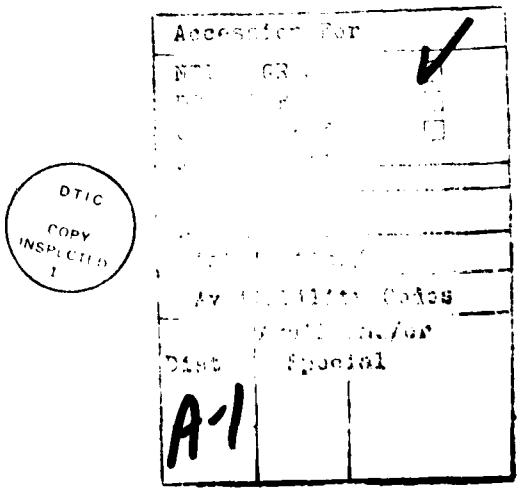
Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fullfillment of the
Requirements for the Degree of
Master of Science

Laura A. Vallado, B.A.
Captain, USAF

December 1984

Acknowledgements

I would like to thank my thesis advisor for his patient encouragement; without it I couldn't have accomplished as much as I did. I would also like to thank my husband, for without him, I wouldn't be seeing this time at AFIT as enjoyable; for without AFIT, I wouldn't have met my husband.



Contents

		Page
Acknowledgements		ii
List of Figures		v
Abstract		vi
I.	Introduction	1
II.	Background	5
	The Grid Intersect Code	8
	The Square Quantization Scheme	10
	The Circular Quantization Scheme	11
	The Grid-Intersect Quantization Scheme	11
	Higher Order Codes	13
	Parallel Quantization Scheme	14
	Triangular Quantization Scheme	16
	Evaluation of Generalized Chain Codes	18
	Quantitative Evaluation of Generalized Chain Codes	20
	Summary	21
III.	Software Overview	22
	Input Parameters	22
	Line Drawing Simulation	23
	Calculation of Grid Intersects	24
	Quantization Node Calculations	25
	Performance Measure Calculations	28
	Summary	30
IV.	The Rotated Drawing Algorithm	31
	The Basic Mathematics	31
	The Algorithm	33
	Summary	41
V.	Changes to the Code	43
	Parallel Quantization	43
	Rotation Algorithm	44
	Summary	49
VI.	Performance Analysis	50

Parallel Quantization Scheme Performance . . . 50
The Rotated Drawing Algorithm Performance . . . 53
Summary 54

VII. Conclusions and Recomendations 55

Bibliography 57

Appendix A: Plots of the Error and Bit Rates 59

Vita 104

List of Figures

Figure		Page
1.	Next Possible Nodes for Ring 1	8
2.	Example Illustrating the Grid Intersect Code	9
3.	Encoding Assignments for Ring 1 Code	10
4.	Square Quantization Scheme	10
5.	Circular Quantization Scheme	11
6.	Grid-Intersect Quantization Scheme	12
7.	Increasing the Curvature of Error	12
8.	Three Rings Around Central Node	13
9.	Generalized Forms for the Link Gate Sets for the (a) Parallel Quantization Scheme and the (b) Triangular Quantization Scheme	14
10.	Parallel Template for Rings 1, 2, and 3	15
11.	Triangular Template for Rings 1, 2, and 3	17
12.	The Incremental Errors in a Coded Line Drawing	21
13.	Rotation of Axes	32
14.	The Rotated Line Drawing	33
15.	A Rotated Line Drawing Which is not a Function	34
16.	The Rerotated Line Drawing with the Rotated Grid Lines	35
17.	The XY and X'Y' Axes Rotated Down Alpha	36
18.	Relationship of the Y = DEL Grid Lines and the X'Y' Axes	37
19.	Relationship of the X = DEL Grid Lines and the X'Y' Axes	38
20.	Finding the Interval Boundaries	41
21.	Peaks of the Sine Curve	45

Abstract

This document implements the parallel quantization scheme for quantizing line drawings. It also discusses the theory behind an algorithm for a rotated drawing quantization algorithm. The algorithm was also implemented during this thesis. Both of these algorithms were analyzed and compared. The parallel quantization scheme was compared with the previously implemented triangular quantization scheme. The rotated drawing algorithm was implemented using the parallel quantization scheme and compared using different rotations.

The results of these computer programs were analyzed using two criterion: compactness and accuracy. Then recommendations were made using these results.

I. Introduction

The technology for quantizing, processing, and storing two-dimensional image data effectively is incalculably essential to many diverse engineering and Department of Defense procedures. The customary method for quantizing two-dimensional image data is through two-dimensional sampling. For high quality resolution, this sampling process requires an excessive amount of memory space and an extensive processing time. Two-dimensional sampling is not efficient for most image data.

A category for which two-dimensional sampling is unrealistic is the line drawing. A line drawing is a two-dimensional image which is wholly composed of thin lines on a contrasting background [2:237]. A class of "regular" line drawings consists of well defined geometric figures with straight lines and relatively few arcs, as in engineering drawings or alphanumeric text [16:1]. "Free form" line drawings represent terrain contours, geographical maps, or other naturally bounded objects [16:1]. When quantizing these objects using two-dimensional sampling, much redundant data is realized, which leads to the need for a more efficient method to quantize and encode these objects.

The family of quantization and encoding schemes which create a more effective means to quantize and encode two-dimensional line drawings is the generalized chain codes. To this date, the analysis of the performance of the generalized

chain codes is performed by primarily quantitative methods.

The objective of this thesis is to extend the efforts of Lt Jones [10], and Capt Christensen [3] by modifying the program which prvides a quantitative analysis of the performance of various chain codes used to quantize specific periodic wave forms. The program was modified to include the parallel quantization method in contrast to the originally implemented triangular quantization method; both of which are discussed in Chapter II. In addition, an algorithm was developed to quantize a SINE wave rotated from 0 to 45 degrees. These two modifications were analyzed using the performance measures of code rate and accuracy discussed in Chapter II.

The accuracy of the quantization is calculated by summing the total area segments between the line drawing and the quantization. This area is then normalized to the total area per the unit length of the line drawing. The code rate is found by calculating the number of bits required to "transmit" the drawing across a channel, depending upon the code. This number of bits is then normalized to the total number of bits per unit length of the line drawing.

The following assumptions were made at the outset of this thesis:

1) The line drawing will not cross over themselves [3:I-3].

2) The grid is located in the right half of a plane of a cartesian coordinate coordinate system such that the vertical grid lines are parallel to the Y-axis [3:I-3].

3) For the rotated line drawing; the line drawing must be a function when rotated back to the X-axis. It does not have to be a function when first drawn on the rotated line.

4) The grid size of the quantizing grid is small enough such that the derivative of the function defining the line drawing has, at most, one zero between any two vertical grid lines [3:I-3].

These assumptions limit the scope of this thesis and facilitate the creation of the line drawings. Defining the line drawings by a function aids in collecting data, since line drawings with the identical characteristics can be created repeatedly for quantization by the different quantization schemes. Furthermore, by utilizing a function to define a line drawing, the error statistics will be simpler to calculate as the statistics for the line drawing will be constant for each period, while only the statistics for the different quantizations schemes will change.

The arrangement of the material in this thesis correlates with the research and design for the problem. Chapter II details the current literature reviewed relating to this area and presents the background material essential for a total comprehension of the topic. It presents a discussion of exactly what line drawings are, many diverse types of line encoding techniques, and an overview of the definitions of the performance measures. Chapter III reviews the previously implemented program developed by Lt Jones used to define line drawings and quantize these line drawings using the triangular quantization method mentioned in Chapter II. Chapter IV

follows the design and theory necessary to create and comprehend the rotated drawing algorithm. Chapter V explains the changes and extensive additions necessary to implement the parallel quantization and the rotated drawing algorithm. The results of the test runs are analyzed in Chapter VI, while any conclusions for this thesis and recommendations for further study are arranged in Chapter VII.

II. Background

A line drawing is one of man's most common means of communication. It is normally a two-dimensional (2d) image which conveys information through its shape, size, manner of interconnection and location of its lines on a contrasting background [2:237]. Of course, there is also symbolic significance in the thickness or the color of the line drawings; or even the texture of the background [4:57]. Since line drawings can include maps, charts, graphs, and printed or handwritten material, they represent an important class of pictoral data used in such diverse areas as cartography, computer graphics, alphanumeric text, and engineering drawings [13:180]. For these uses, it is necessary to store these drawings for later retrieval or transmission; therefore, the drawings need to be quantized into computer readable data.

Since a line drawing is a form of communication; processing line drawings is in reality processing information [15:31]. Processing line drawings refers to two different operations: extracting 2d line structures from line drawing images, and preparing the line structures to extract information of interest [4:60]. This thesis will deal with preparing the line structures to extract information of interest from them. There are three terms which will be used frequently, so they will be defined now. The first term is "image". According to Freeman, an image is a natural visual object which has

2d spatial variations of color, brightness, or both. It is objectively sensed by the human eye; although it does not depend upon an assignment of meaning for definition. Examples of images include: paintings, photographs, and printed text. Another term which should be defined is a line structure. A line structure can be geometrically defined using points, line segments, and curved segments in Euclidean space. These points and segments do not have to be joined, but they can be defined in an appropriate coordinate system. The third term, line drawing, is used to illustrate an image which will convey information about a 2d line structure [4:57-60].

An image can visually represent a line structure, which is a highly subjective process; dependent upon the observer, his experience, the time, the place, and the context of the information [4:58]. Therefore an image can become a line drawing in the mind of the viewer if that is his perception [4:60].

There are three ways a line structure can originate; each will affect the way it is processed [4:60]. First, it may be defined abstractly in a geometric sense, where a line drawing will serve as a model for the line structure [4:60]. Tracings constitute the second source of line structures. A tracing in a plane leads to 2d line structures, which can be represented by line drawings [4:60]. The third source of 2d line structures is images, but in this case, the line structure is the model for the image [4:60].

There are many problems with processing line structures, which can be grouped into four major categories. These categories are based upon the input and ouput of the operations. In the first, analysis, a line structure is input and its characteristics are produced. In the second, synthesis, a line structure is generated with certain characteristics and is then displayed by labelled line drawings. Manipulation is concerned with the transformation of line drawings, while the final category, pattern recogniton, is concerned with the problems of classifying line structures [4:60].

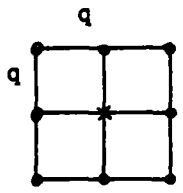
Of course the problems of processing line structures do not outweigh the need to have ready and rapid access to reams of 2d data. This leads to the requirement of efficiently quantizing, processing, and storing this data. Line by line scanning, as done for general image processing, is a common method of quantizing 2d images. This method requires vast amounts of computer memory and time [10:1]. The scanning technique can be made more efficient by using run-length coding. An alternative approach for quantizing and encoding line drawings is the grid-intersect method.

The grid-intersect quantization method superimposes a cartesian grid over the line to be quantized. The line to be quantized is referred to as a grid-based line drawing, and the horizontal and vertical grid line intersects are referred to as nodes [9:5]. For the purpose of this thesis, a line drawing is quantized when all of the nodes closest to the

grid-intersects are selected. If two successive grid-intersects encode into the same node; that node is only selected once [9:5]. A chain is formed when successive nodes are connected by straight line segments [9:8]. Each node is encoded by assigning a three digit binary number, which represents its relative position from the previously selected node [10:3]. This sequence of quantizing and encoding the line drawing is called chain coding.

The Grid Intersect Code

The simplest form of the chain code is the grid-intersect code or the 8-point chain code. The 8-point chain code forms the basis for the family of generalized chain codes [10:3]. As the name implies, this code allows the selection of 8 next possible nodes, which form a square around the original node called the ring. Figure 1 illustrates a typical grid with the current node and the ring of next possible nodes around it.



x - current node

o - next possible node

q - grid size

Figure 1. Next Possible Nodes for Ring 1

The distance along the horizontal and vertical grid lines between the noes is of length q, then the distance

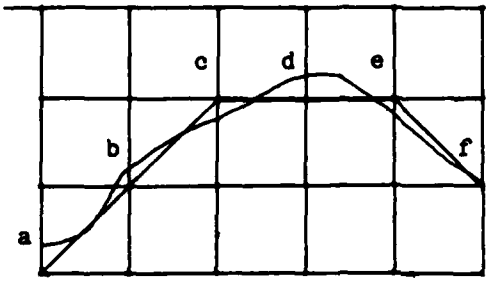


Figure 2. Example Illustrating the Grid Intersect Code

between the current node and the next nodes is either q or $\sqrt{2}$ q [5:1]. When the nodes lie on a square of side 2q and are centered around a point, the square is called a ring 1 [5:1]. Figure 1 also illustrates a ring 1 code grid. To encode a line drawing, locate the first grid-intersect, find the node closest to it, and encode this node as the starting point. *Now find the next intersect on ring 1 around the starting* point, and encode this as the next node. This is continued as illustrated is Figure 2 until all of the grid-intersects have been encoded.

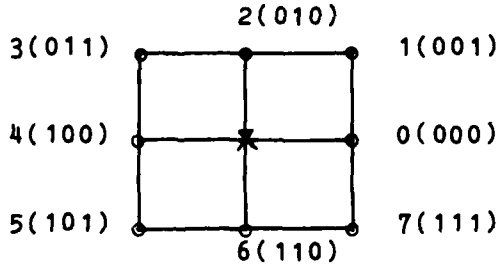


Figure 3. Encoding Assignments for Ring 1 Code

The necessary binary encoding procedure is to associate an octal integer (3 bit binary number) with each node of the

ring as shown in Figure 3 [15:5].

This assignment of the octal integers is arbitrary, but it must be kept consistent for encoding and decoding the nodes. The grid nodes are encoded by representing the relative position from the original node with one of the eight octal (binary) numbers. For the example in Figure II-2 the line drawing would be represented by the chain 11007.

The Square Quantization Scheme

The square quantization scheme is the easiest scheme to use. To set the grid up for the line quantization, a square is drawn at each node. This square has a diagonal of length T, the length of the lines between the nodes, the sides are of length $\sqrt{2}$ T, as illustrated below.

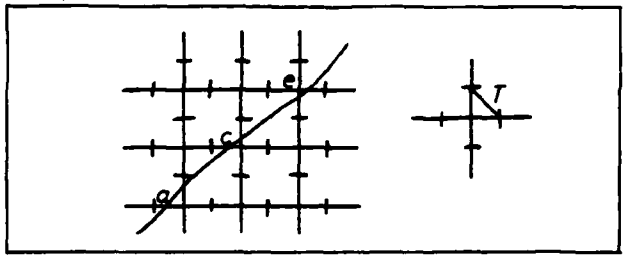


Figure 4. Square Quantization Scheme

The diagonal of the square runs along the lines extending from the node. If the line passes through the square for this node, then this node is used as the next node for the chain code. It can be seen that if the sides of the square were any shorter, the line could conceivably pass by all of the eight

possible next nodes in the chain code.

The Circular Quantization Scheme

The circular quantization scheme is much like the square quantization scheme, except instead of a square centered at the node, a circle of radius T/2 is centered at the node as illustrated in Figure 5.

When a line is placed on the graph, if it passes through the circle, then that node is encoded into the chain code. It can also be seen that if the radius of the circle was not

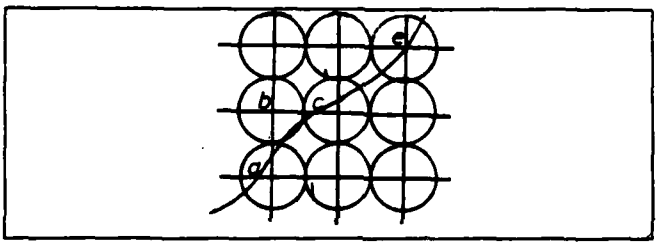


Figure 5. Circular Quantization Scheme

2/T, that the next node encoded would not be one of the eight allowable nodes in the chain code. Conversly notice that by making the radius 2/T (and for the square, the diagonal T), that the edges are tangent to each other (intersect at the corners); therefore, there is no way for the line to be encoded incorrectly.

The Grid-Intersect Quantization Scheme

In the grid-intersect quantization method, a square is also centered at the node. The sides of the squares meet

between the nodes so all of the area between the nodes is covered as illustrated in Figure 6.

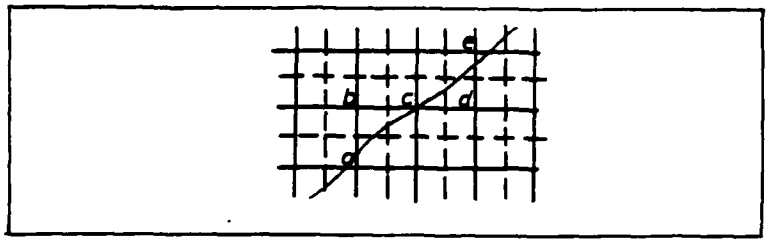


Figure 6. Grid Intersect Quantization Scheme

To select next nodes, the line on the graph is followed and whenever it passes into a square, that node is the next node. With this scheme there is no chance that a line will miss a square and select a next node that is not one of the eight allowable next nodes. A short comparison of the first three schemes might help to show the advantages of each.

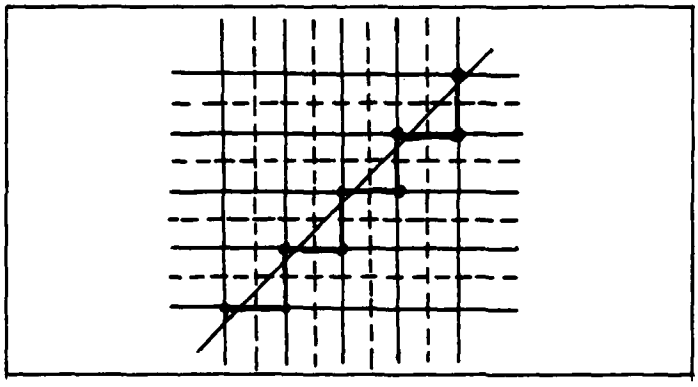


Figure 7. Increasing the Curvature Error

To compare the three schemes, consider the three previous figures [13:181]. A node could be left uncoded by using the square or circular approach because the squares and circles do not cover the total area. In fact, there will

still be error in the grid-intersect scheme because the encoding might place more of a curve upon the line than it originally had, as can be seen in Figure 7.

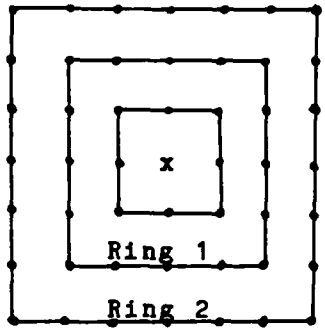


Figure 8. Three Rings Around Central Node

Higher Order Codes

To discuss the next two quantizing schemes, a few more terms need to be defined. Schemes 4 and 5 use higher order codes, which are just chain codes with more permissible next nodes. These next nodes are on rings which are just squares of the grid which contain all the nodes around that center node. The first ring contains eight nodes and is essentially the same as illustrated in Figure 1. Moving out from the center one more grid space, a ring can be created containing sixteen nodes; this is ring 2. Then, move out one more grid space from the center to encompass twenty-four nodes in ring three [6:1]. Figure 8 illustrates rings one to three [6:1]. The encoding process for the higher order codes is facilitated through the use of link gate sets. Different encoding

schemes are developed by defining new link gate sets. Two schemes to be discussed in this thesis are the parallel quantization scheme and the triangular quantization scheme. General forms of the link gate sets for each of these schemes are shown in Figure 9.

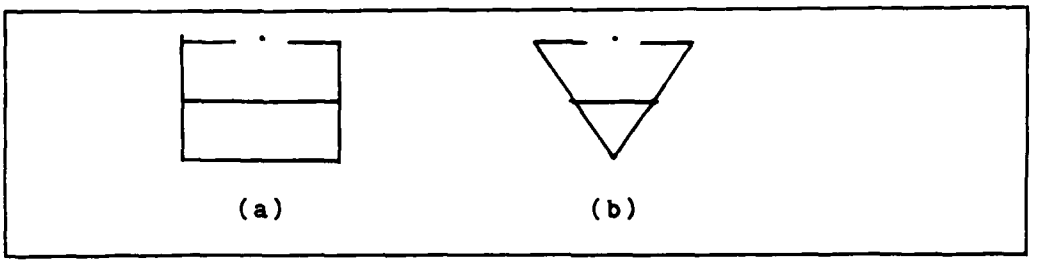
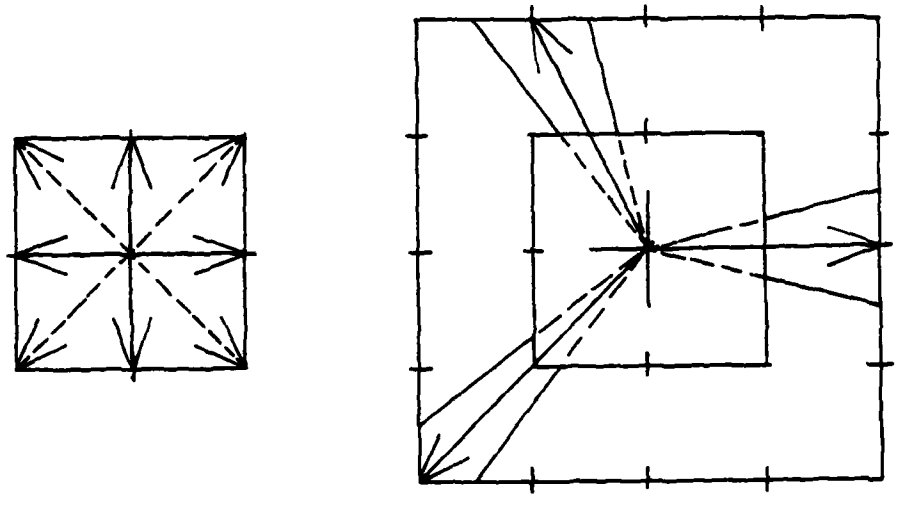


Figure 9. Generalized Forms for the Link Gate Sets for the (a) Parallel Quantization Scheme and (b) Triangular Quantization Scheme

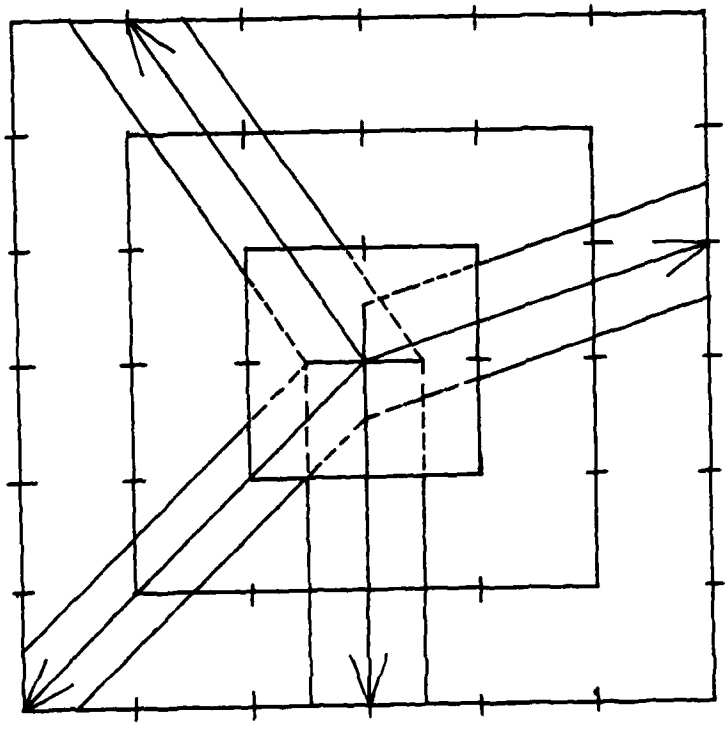
Parallel Quantization Scheme

The LGS for the parallel quantization scheme are defined by choosing the highest-order-ring contained in the code to be used and finding the midpoints between every node on that ring. Next find the midpoints of the four solid lines extending from the current node to the sides of ring 1, as illustrated in Figure 10(a). Then draw two lines from the pair midpoints beside each node to the corresponding midpoints found inside ring 1. These lines will either intersect the interior on a midpoint or a node. The two parallel lines for each node are called the midpoint lines. This process is repeated for each ring in the code. The set of all possible



(a) Ring 1.

(b) Ring 2.



(c) Ring 3.

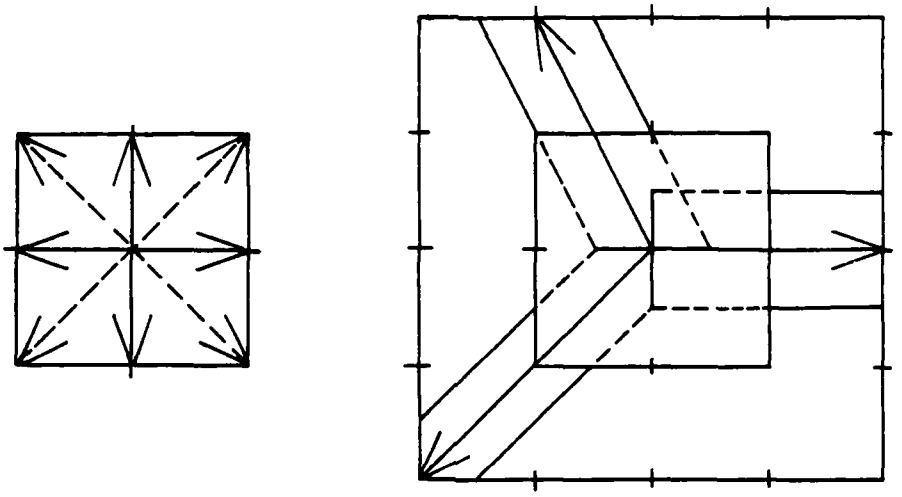
Figure 10. Parallel Template for Rings 1, 2, and 3.

LGS for a code is called the "template" for that code [16:15].

To quantize the line, a search begins for the highest-order-ring link for which all link gates intersect the curve. The template is placed on the center node, and all intersection points of the curve with all the rings are located. If all of the intersections of the highest-order-ring are contained in the LGS then the node for that LGS is selected for encoding; if not, the next lower ring is checked until this criterion is met. The LGS of the lowest-order-ring will always meet the quantization criterion [16:15]. This method of quantizing a line will give a better code than the first three methods because it is extended to the outer rings, giving less bit quantization if the line drawing continues basically in the same direction. Figure 10 illustrates the templates associated with rings 1, 2, and 3 using the parallel quantization scheme.

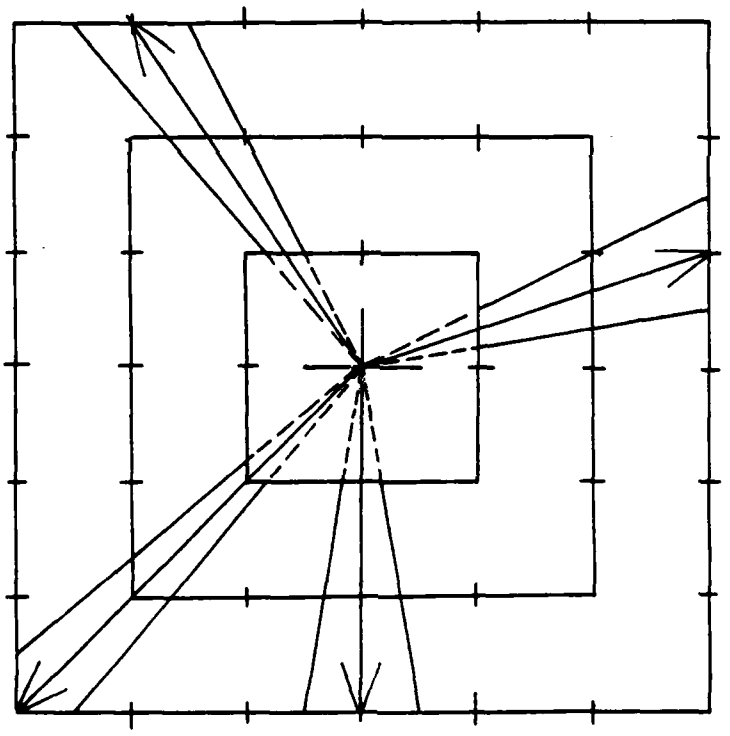
Triangular Quantization Scheme

Another method in the same line as the parallel quantization scheme is the triangular quantization scheme. The LGS for the triangular quantization scheme are also formed by finding the midpoints of all the node pairs on the highest-order-ring of the code. Lines are then drawn from the midpoints to the center of the ring. These lines are also referred to as the "midpoint lines". These lines cut parallel



(a) Ring 1.

(b) Ring 2.



(c) Ring 3.

Figure 11. Triangular Template for Rings 1, 2, and 3.

segments out of each ring to form the LGS for the highest-order-ring. This process is repeated for each ring in the code. This will form a template for triangular quantization scheme [16:19].

The procedure for quantizing the line is similar to the parallel scheme; first, the template is placed over the last node to be encoded, then the LGS for the highest-order-ring is checked and if it's not inside then the next is checked and on down. The difference lies in the fact that the template does not include all of the space the line could pass through. So if the lowest-order-ring is reached and cannot be quantized, there are two alternatives. One is to find the first intersection between the line drawing and the grid lines, to locate the node nearest the intersection, and then to quantize and encode it as the next node. The other alternative would be to find the first intersection of the line drawing and the lowest-order-ring, to select the node nearest to the intersection, and then to quantize and encode it as the next node [3:II-12]. Figure 11 illustrates the templates associated with the triangular quantization scheme for rings 1, 2, and 3.

Evaluation of Generalized Chain Codes

There are six suggested major criteria to evaluate the effectiveness of a quantization scheme. These criteria are: 1. compactness; 2. precision; 3. smoothness; 4. ease of

selective access; 5. ease of processing; 6. ease of encoding and decoding [8:315]. Depending upon the application of the quantized data, these criteria are given different weights.

Compact quantization schemes require fewer bits to store or transmit a given line drawing. However, tradeoffs are made with the other five criteria to achieve high compactness, and compactness will lead to increased complexity in encoding, decoding, and processing [8:315].

High precision occurs when the area error of a line quantization is very small. High precision will create extremely low compactness, since precision is highly dependent upon the grid size. Therefore, if a very small grid size is used in relation to the radius of curvature for the line drawing, *high precision* will result, but with a cost for compactness.

The ease of selective access refers to the relative speed with which any particular portion of a quantized line drawing may be accessed [8:316]. Normally the access criterion creates a decrease in compactness and therefore increases storage and transmission requirements [8:316].

The simplicity and speed of the quantization algorithms defines the ease of processing, which is extremely dependent upon the application of the line drawing. If the line drawing is only stored, then the ease of processing is less important than the compactness criterion; however, if the data is processed frequently, then the ease of processing is more

critical.

The ease of encoding and decoding becomes extremely important if large amounts of data are being processed. For small quantities of data which require extensive processing, the ease of processing criteria takes precedence [5:1].

Quantitative Evaluation of Generalized Chain Codes

This thesis effort will use two performance criteria to quantitatively analyze the performance of the generalized chain codes. The two criteria which will be measured are the precision and compactness criteria or in information theory terms, distortion and code rate.

Distortion measures how closely the digitized line drawing approximates the original line drawing; therefore, distortion will be used to measure the accuracy of the quantization. The distortion calculation will be the normalized measure of area between the original line drawing and its quantized approximation [1:979].

The normalized area error is calculated by summing all of the incremental area errors lying between the original line drawing and its digitized representation, and then dividing this sum by the length of the original drawing in terms of grid size. Figure 12 illustrates the incremental area errors in a typical line drawing encoded by the 1-code.

The total area error must be normalized by the length of the quantized line to find the normalized area error; the

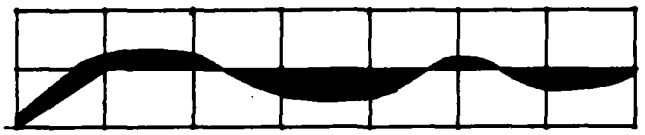


Figure 12. The incremental errors in a coded line drawing

distortion measure. A detailed discussion of the distortion measure being used is found in references 1 and 2.

The code rate performance measure refers to the number of bits required to quantize a length q of the line drawing. This is obtained by determining the number of bits required to represent all of the possible next nodes. For example, if a 1-code were being used, 3 bits would be required and for a 1,2-code, 5 bits would be required to represent the next possible nodes. Using the 1-code to quantize the line drawing in Figure 12, the code rate performance measure is defined as $R = 3\ COS(0)/nq$. A detailed discussion of the code rate can also be found in references 1 and 2.

Summary

This chapter dealt with the concepts of line drawings and methods of processing line drawings. Types of grid-intersect codes and generalized chain codes were also discussed. Five methods of quantizing and encoding line drawings were presented as well as a set of criterion for evaluating the effectiveness of the quantization methods.

III. Software Overview

This chapter will present a brief overview of the software algorithm designed and implemented by Lt. Keith Jones (10:20) and modified by Capt. Christensen (3:III-1). The developed algorithm was designed specifically for the quantization of SINE and circular waves. The triangular scheme described in Chapter 2 was used to quantize the curves. This software overview will briefly present the input parameters, the subroutines which simulate the line drawing, the calculation of the intersect points, the triangular quantization of the nodes, and the calculation of the performance measures.

Input Parameters

The input parameters required for the execution of the program are contained in an input file, of which the last two characters are 'in'. These parameters are read in at the beginning of the program, and are not changed after that. The input parameters are as follows:

1) LFN: this is the name of the file to which all of the output data is written.

2) NF: tells whether the function to be implemented on this run is a SINE or circular wave.

   If NF = 1 use the SINE function
      NF = 2 use the circular function

3) A: this is the amplitude of the SINE function (the amplitude of the circular function is one-fourth the period).

4) PER: this is the period of the function.

5) THETA: the initial phase point of the function.

6) DEL: the grid size.

7) NPER: number of periods of the function to be calculated.

8) NLC: number of ring levels in the code.

9) NRL(L): the ring-number at level L of the code, which is an array, L = 1,2,...,NLC.

Input parameter (8) specifies the number of rings in the quantizing code. For example, the (1,3) code has two rings, so NLC would be 2. Now input parameter (9) describes the rings of the code. So, for the (1,3) code, at level L = 1, NRL would be 1, but at level L = 2, NRL would be 3.

Line Drawing Simulation

For the line drawing to simulate a function, a few FORTRAN subroutines were written to define the simulation. The FORTRAN subroutine to define the function (or the line drawing) was F. The subroutine F was used no matter whether a SINE wave or a circular wave (NF = 1 or 2) was being implemented. To find the derivative of the SINE or circular wave, the FORTRAN function DF was written, and the inverse of

either curve is found by the subroutine FINV. A thourough discussion of these functions is given in Lt Jones' thesis (10:22-25).

Calculation of Grid Intersects

The calculation of the grid intersects for the SINE wave and the circular wave is accomplished by the subroutines FIP and CIBD.

FIP FIP is called by the main routine to calculate the grid intersects for the functions. There are four parameters passed to this subroutine: TSTART, NIPST, NLIMIT, and JFIN. TSTART is the location of the first grid intersect to be calculated, which is the beginning x-value of the function the first time through (which is set to zero), or the place that FIP left off when it filled the array the last time through. NIPST is the starting value for the pointer to the grid intersect array and is set to one the first time FIP is called. NLIMIT is the largest subscript for the pointer and is calculated in the main routine. NLIMIT is set equal to one less than the dimensions of the array minus twice the amplitude of the function being quantized (10:26). JFIN is a flag which is set to one when the subroutine is entered. JFIN indicates whether all of the grid intersects were calculated before the NLIMIT is reached. If NLIMIT is reached, JFIN is set to zero to indicate the subroutine has not completed the

calculations. The FIP subroutine only calculates the vertical grid line intersects. FIP does call the subroutine CIBD when a horizontal grid line intersect needs to be calculated. FIP also uses the functions F and DF and the subroutine FINV to calculate the vertical grid line intersects (10:26-30).

CIBD The CIBD subroutine calculates the horizontal grid line intersects for the line drawing over a specified interval in which the line drawing is strictly non-increasing or non-decreasing. The inputs to CIBD are Y1 and Y2 the x-values of the end points of the interval, DY1 and DY2 the derivatives of the function at Y1 and Y2, respectively, and T1, the last vertical grid line calculated. CIBD returns the updated peak/zero crossing counter, NQCF. CIBD also uses the function, JSGN, which returns a -1, 0 or +1 for a negative, zero, or positive number. The subroutine FINV is also called by CIBD to find the x value for a particular horizontal grid line.

Quantization Node Calculations

The section of the program which calculates the quantization nodes contains five subroutines: GERD, FNSRI, FNGN, FAR, and ANGLE. These subroutines are independent of whether the line drawing to be quantized is rotated or not. They are, however, dependent upon the triangular quantization scheme, as that is the scheme they implement. The subroutine GERD is the main subroutine calling the other four subroutines as

necessary.

GERD There is one input to GERD so that at the end of the intersect array, JFIN is used to see if there will be more intersects, and if there will be more, the end of the node array is setup for the next call to GERD. NSV is another pointer to the location in the grid intersect array where the routine stops. GERD calculates the first quantization node for the function and then proceeds through the grid intersect array eliminating redundant grid intersects. Then FNSRI is called to calculate the subscript of the grid intersect on the current ring. It then calls FNGN to calculate the node closest to the grid intersect. After the node is calculated, the subroutine FAR is called to calculate the valid angle range for the triangular quantization. *Subroutine ANGLE is* then called to check the angles for all of the points between the two nodes, and of they are all good, the node is stored. If not, the next smaller ring is tried, on down to the smallest (10:32-34).

FNSRI The two inputs to subroutine FNSRI are NSV, the pointer to the current grid intersect and NRS, the current ring level. These two inputs are used to find the subscript of the first grid intersect, after grid intersect NSV, which lies on ring NRS. This subscript is stored and passed back by the variable NSRI, or a zero is passed back in NSRI if no intersect was found (10:34).

FNGN The pointer NSRI is input to subroutine FNGN. This subroutine calculates the coordinates of the closest grid node to the grid intersect whose subscript is NSRI. The subroutine returns the x-coordinate and y-coordinate values in the variables XNGN, and YNGN (10:35).

FAR The subroutine FAR has four inputs: NSV, the pointer to the current grid intersect; NRS, the current ring level of the code; XNGN, the x-coordinate of the next node; and YNGN, the y-coordinate of the next node. This subroutine calculates the validation angle's upper and lower bounds, and to do that, the location of the node on the ring must be found. After the location is determined, the coordinates of the points one-half of the grid size on either side of the node are calculated. The subroutine ANGLE is then called to calculate the angle for the upper and lower bounds. These bounds are returned through the variables ALB and AUB, and a correction factor for the lower bound being in the fourth quadrant, CF (10:35-36).

ANGLE This subroutine calculates the angle between the line segment from the center of the ring to the point (X,Y) and the x-axis, where the origin is defined to be at the center of the ring. The pointer to the current node, NSV, is input, also the point (X,Y). These values are normalized, and the angle is then calculated and returned via the variable ANG (10:37).

Performance Measure Calculations

The performance measures are calculated through the use of eight subroutines which are: CAE, CAESL, FDE, DFDE, ZERO, FDLI, and SPGQI. The subroutines calculate the area error and path length of the line drawing for an arbitrary interval.

CAE This subroutine calculates the area error between the line drawing and the linear interpolation of the quantization by subdividing the integral into subintervals whose limits are defined by the vertical grid lines within the limits of integration. Variables A and B are input as the initial limits on the interval, and the summed value AE is the output area error (10:38-39).

CAESL This subroutine accepts the interval inputs A and B and calculates the area error over the subintervals defined in CAE. In this subroutine, the interval is divided into subintervals over which the integrand is strictly positive, or strictly non-positive. This subroutine uses the function FDE which defines the integrand of the area error integral and DFDE which gives the derivative of FDE. Depending upon what this integrand and its derivative do, the subroutines ZERO and SPGQI are called using many different parameters for many different cases. The output of this subroutine is also area error, AE (10:39).

FDE This function defines the integrand of the area error

integral, given a specific value, X, to evaluate it at. This function calls function F for the function of the line drawing at this given point. This function is also an input to the subroutines ZERO and SPGQI.

DFDE This function defines the derivative of the integrand of the area error integral, given a specific value, X, to evaluate it at. This function calls function DF for the derivitave of the line drawing at this given point. This function is also an input to the subroutines ZERO and SPGQI.

ZERO This subroutine calculates the zero in any function between specified limits. Only one zero can be found. This subroutine uses the modified regula falsi algorithm, which Lt Jones describes (10:40), to calculate the zero. The inputs to this subroutine are the function for which the zero needs to be found (either DFDE or FDE) and the endpoints of the interval, A1 and B1. It returns the variable XZERO which is the x-coordinate of the zero point (10:39).

CLI This subroutine calculates the path length between the specific input limits A and B. This interval is then subdivided into subintervals of widths equal to one-tenth of the grid size to ensure accuracy. This subroutine calls SPGQI to get each incremental length and the sums them up to be output through the variable, VLI (10:40).

SPGQI This subroutine calculates the integral of a function

over a specified interval using six-point qaussian quadrature integration. Ths function to be used is input (either FDE, DFDE, or FDLI) along with the interval boundaries A and B. The integration is summed up and output via the variable SUM (10:40).

FDLI This function defines the integrand of the path length integral. The value X, for which it is to be defined, is input and is uses the function DF to calculate the returned integrand.

Summary

This chapter gave a brief overview of Lt Jones' software to be modified is this thesis. It discussed the input parameters, the line drawing simulation, the calculation of the grid intersects, the quantization node calculations using the triangular quantization scheme, and the performance measure calculations.

IV. The Rotated Drawing Algorithm

This chapter will detail the theory and steps for the rotated drawing algorithm. Implementation of this software algorithm will be discussed in the next chapter. First, the background mathematics will be described, then expanded into the steps necessary to accomplish the rotated drawing algorithm.

The Basic Mathematics

To begin with, we have a line drawing whose equation is:

```
Y = A * SIN((( 2 * PI / PER ) * X) + THETA)
```

If this line drawing is rotated around the origin of the xy plane an angle of *ALPHA (not to exceed 45 degrees), then* we have a rotated SINE curve which may not be a function.

Since this curve may not be a function, the problem is finding an equation for the rotated SINE curve. The previous thesis by Capt. Christensen (3:IV-3) tried using a Lagrangian interpolating polynomial, resulting in too much computer time and extremely inaccurate results. Correcting this algorithm would result in using even more computer time.

After an extensive literature search, an algorithm was discovered which used the principal of rotation of axes (12:66). The formulas to accomplish this rotation are:

```
X = X' COS(A) - Y' SIN(A)
```

$$Y = X' \text{ SIN}(A) + Y' \text{ COS}(A)$$

where X', Y' are the coordinates of the points in the 'new' plane, X, Y are the points in the 'old' plane, and A is the angle of the plane's rotation, as illustrated in Figure 13.

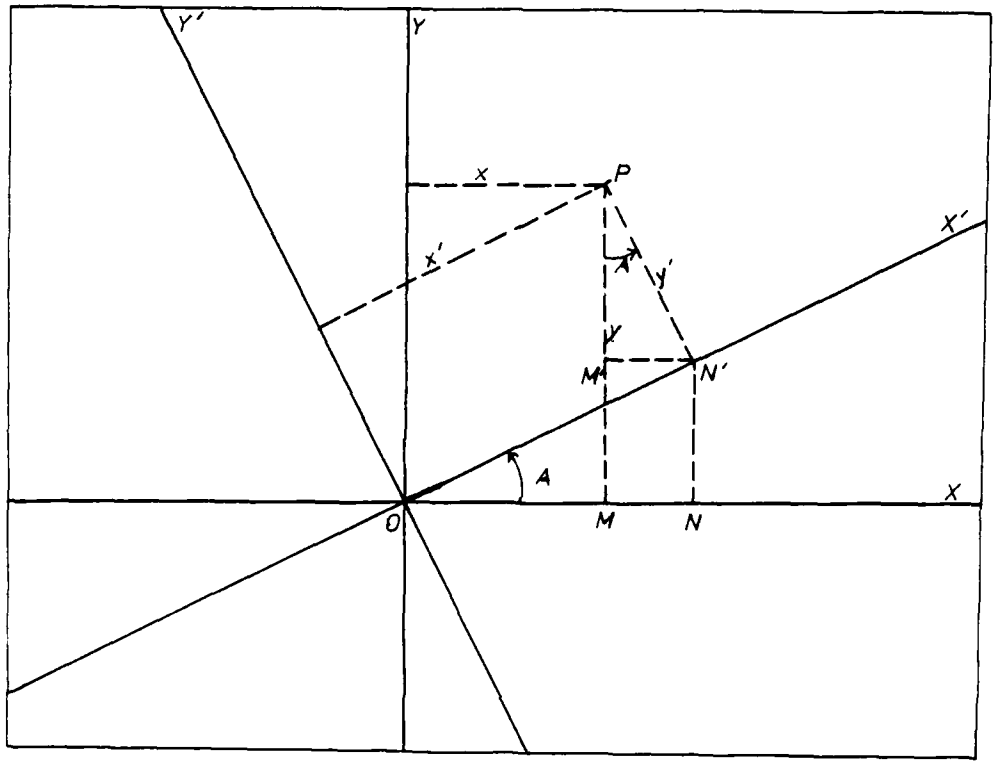


Figure 13 Rotation of Axes

To derive these equations for X and Y in terms of X', Y', and A, proceed as follows:

$$X = OM = ON - MN$$
$$= X' \text{ COS}(A) - Y' \text{ SIN}(A) \qquad \text{and}$$
$$Y = MP = MM' + M'P = NN' + M'P$$
$$= X' \text{ SIN}(A) + Y' \text{ COS}(A) .$$

This gives us the formulas for the rotation of the axes through an angle Alpha.

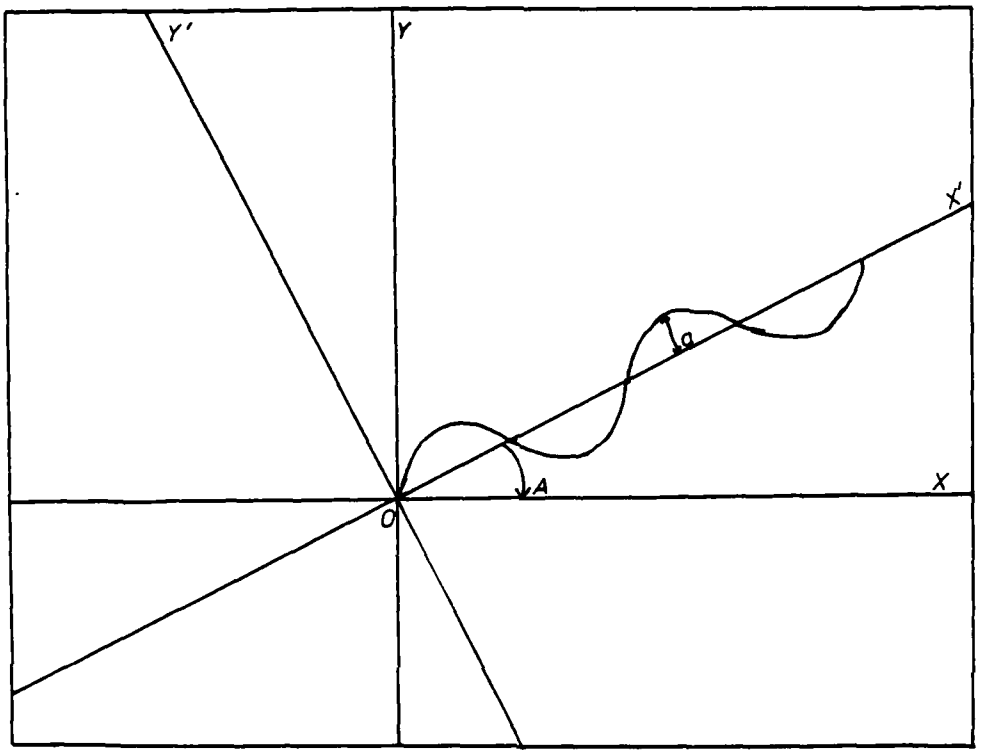


Figure 14. The Rotated Line Drawing

The Algorithm

Consider the rotated line drawing which is illustrated in Figure 14. This SINE curve has an amplitude of 'a' and its x axis is rotated an angle of Alpha. Now, consider this drawing with a different amplitude or period. The drawing in Figure 15 illustrates a curve with the same period but a greater amplitude. This drawing is not a function, but there is a way to make it a function and still get the XY grid intersects.

Consider for a moment what would happen if we rotated both the XY and X'Y' planes back an angle of A. The planes would still have the same relative position to each other, but a different perspective of them would be seen (Figure 16). Now consider the grid lines of the XY plane, which is rotated down an angle of Alpha. These grid lines have also been rotated down Alpha, yet since they are lines, their equations can easily be found in the X'Y' plane. With this second rotation, the rotated line drawing is now a function in the X'Y' plane, and using the XY grid lines now defined in the X'Y' plane, the intercepts can be found.

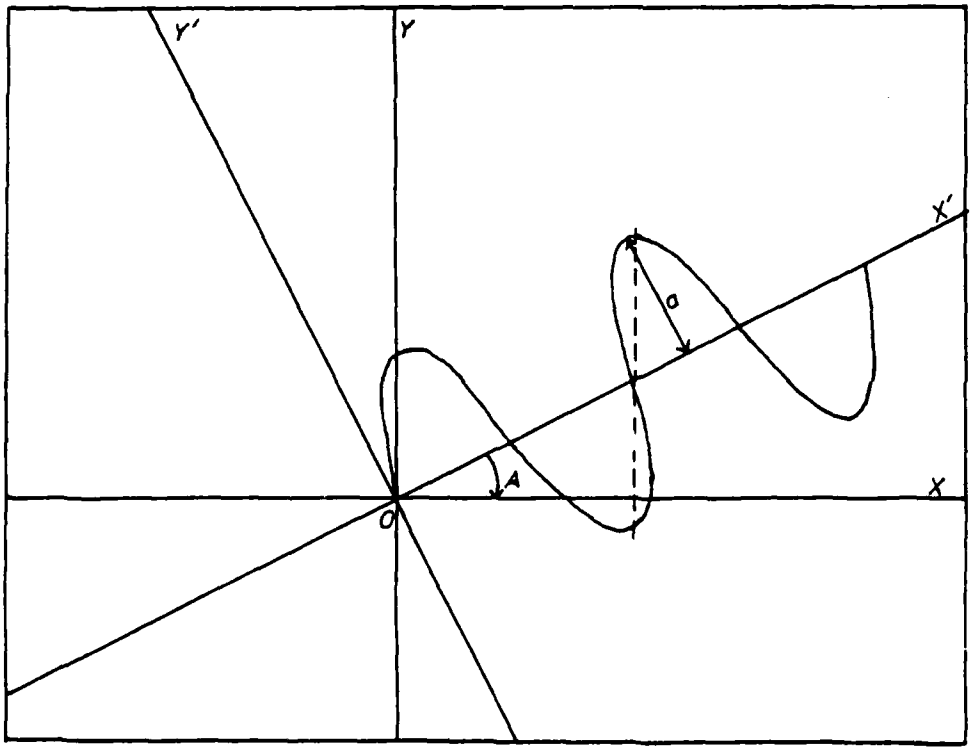


Figure 15. A Rotated Line Drawing which is not a Function

To find these intercepts, a few preliminary numbers must be found. The slopes (MX and MY) must be found, and also the y'-intercepts of every XY grid line must be found. This sounds like a considerable amount of data, but it ends up being rather easy once the first y'- intercept is found.

But first, the slopes must be calculated. Figure 17 shows the two sets of axes rotated down Alpha. To find the slope of the X axis (and all the lines where Y = a multiple of DEL), proceed as follows:

Slope = Mx = Rise / Run = y' / x'

also

TAN(Alpha) = Opposite / Adjacent = y' / x'

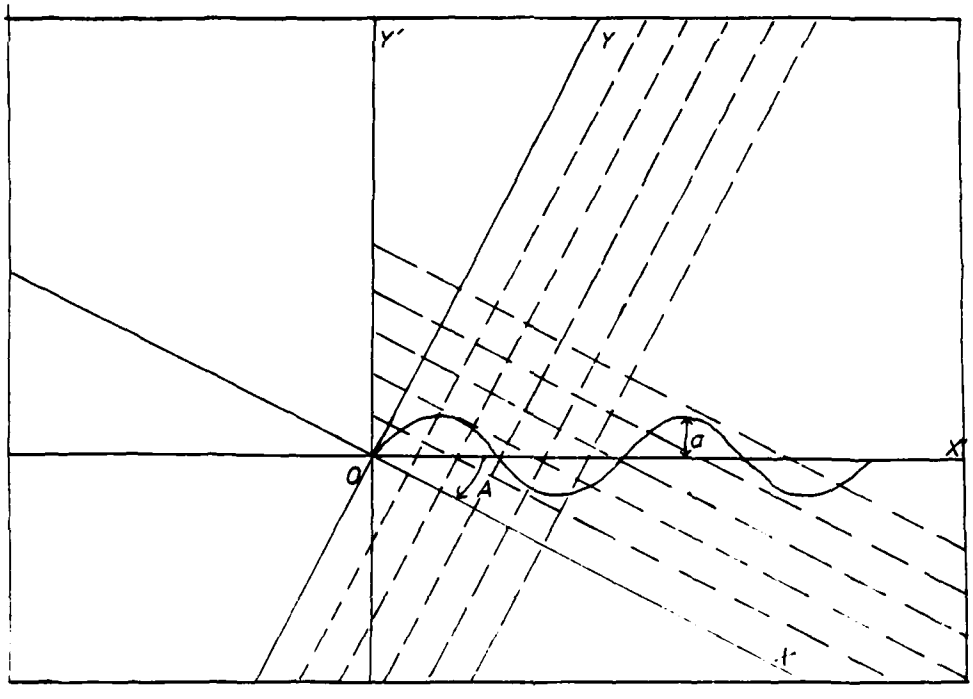


Figure 16. The Rerotated Line Drawing with the Rotated Grid Lines.

where X' is DEL (the grid size), since DEL is the basic unit of measure in all of the calculations throughout the simulation program. Therefore

$$MX = TAN(A)$$

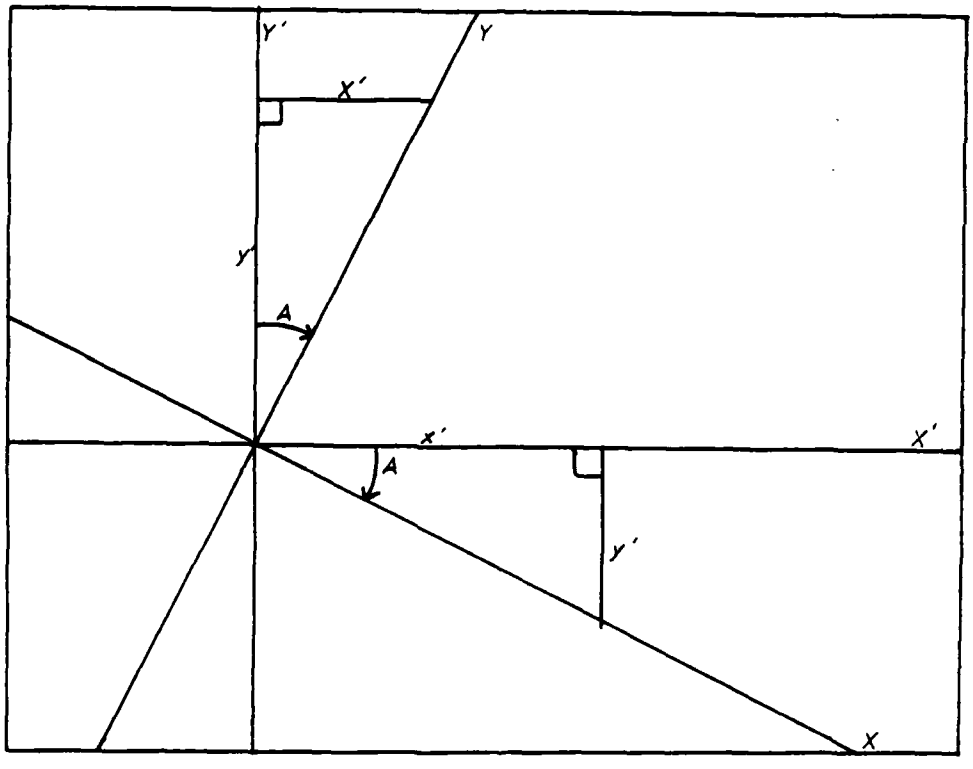


Figure 17. The XY and X'Y' Axes Rotated Down Alpha.

We also know that two lines at right angles have slopes equal to the inverse of the negative of the other, so:

$$MY = -(1 / MX).$$

Now, to find the y'-intercept of the y = n DEL lines. These lines have a slope of MX, which means their equations are:

$$Y' = MX(X') + BX$$

To find BX, look at Figure 18. Notice since the X-axis and the line y = DEL are parallel, and that the X'-axis cuts both lines, angle A equals angle B (11:153). Also since angle B

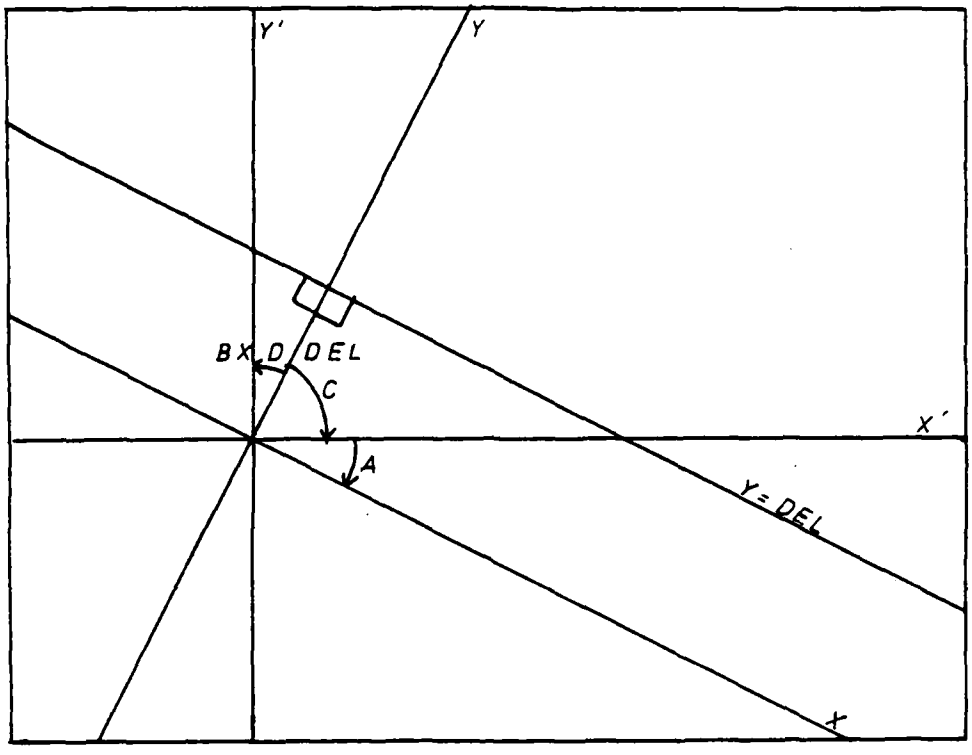


Figure 18. Relationship of the Y = DEL Grid Line and the X'Y'-Axes.

and angle C are the other two angles in a right triangle, they add up to 90 degrees (11:172). Therefore, since angle C and angle D add up to 90 degrees (11:126), angle D = angle B (11:132), which is Alpha degrees. The segment between angles C and D is also known (the Y axis) has a length of DEL since it is the perpendicular distance between the X axis and the

line y = DEL.

Knowing angle D and its adjacent side, BX can be found.

D = Alpha degrees

adjacent side = DEL

COS (D) = DEL / BX

BX = DEL / COS (D)

BX = DEL / COS (Alpha) .

The benefit of finding BX is that the Y'-intercept of each successive line is just a multiple of the number of DEL's out the line is. So if the line is y = 15 DEL, the Y'-intercept, BX is 15*BX.

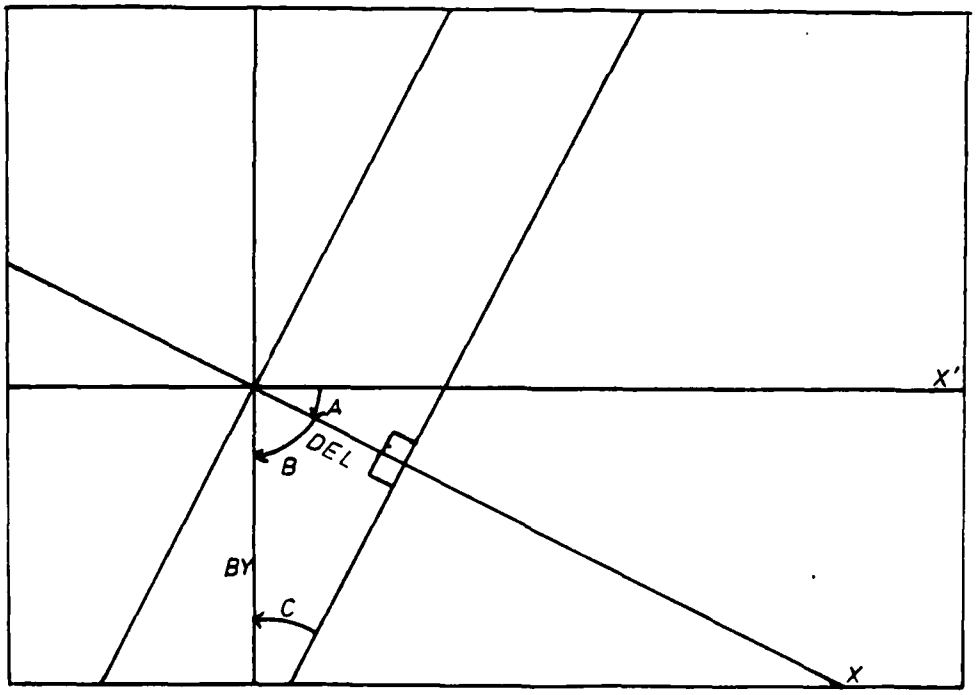


Figure 19. Relationship of the X = DEL Grid Line and the X'Y'-Axes.

Now that BX has been found, BY should be found for all the lines X = n DEL. These lines have a slope MY, so their equations are of the form:

$$Y' = MY\ (X') + BY$$

To find BY, examine Figure 19 and notice that angles A and B add up to 90 degrees (11:126). But angles B and C also add up to 90 degrees (11:172); therefore, angle C equals angle A (11:132). Also, the segment between angles A and B (the X axis) has a length of DEL since it is the perpindicular distance between the Y-axis and the line X = DEL.

Knowing angle C and its opposite side, BY can be found.

$$C = \text{Alpha degrees}$$

$$\text{opposite side} = DEL$$

$$SIN\ (C) = DEL / BY$$

$$BY = DEL / SIN\ (C)$$

$$BY = DEL / SIN\ (Alpha).$$

The same thing applies to BY as to BX. The Y'-intercept of each successive line is a multiple of the number of DELs out the line is.

Now that the equations for the grid lines have been found, the grid intercepts should be found. There is no way to find the unique solution for these two equations:

$$Y' = a\ SIN\ (2*PI / PER\ (X') + THETA)$$

and $Y' = MX\ (X') + BX$ or

$Y' = MY\ (X') + BY$.

Therefore, numerical analysis methods must be used to find the intersections between the two equations.

The chosen method is called the half interval search (14:83). This method requires that an upper and lower bound for the intersection be given for the search interval. These two numbers are then summed together and divided by two to give the center of the interval. The center is now the x value, and is inserted into both equations to find theis appropriate y values. The two y values are then subtracted from eacn other. If this difference is negative, the center x value becomes the new upper bound; if it is positive, the center x value becomes the new lower bound. A new center x value is calculated, and everything starts again. When the absolute value of the difference between the two y values is less than 10 exp -8, the solution is close enough for double precision considerations and it is accepted. This search method seems to take between 15 and 25 iterations when given good boundary values.

Now, to calculate boundary values. The easiest way to get boundaries is to calculate the SINE curve at its peaks, as illustrated in Figure 20. Each X'Y' coordinate is then rotated to find what its coordinate is in the XY plane. Now, the absolute integer distance between the x values (or y values) of the two peaks is the number of grid intercepts. If the curve crosses the X-axis, one must be added to the y distance; as also if the curve crosses the Y-axis, one must

be added to the x distance since the axes are grid lines also. This is all that is needed to calculate the grid intercepts between the two peaks.

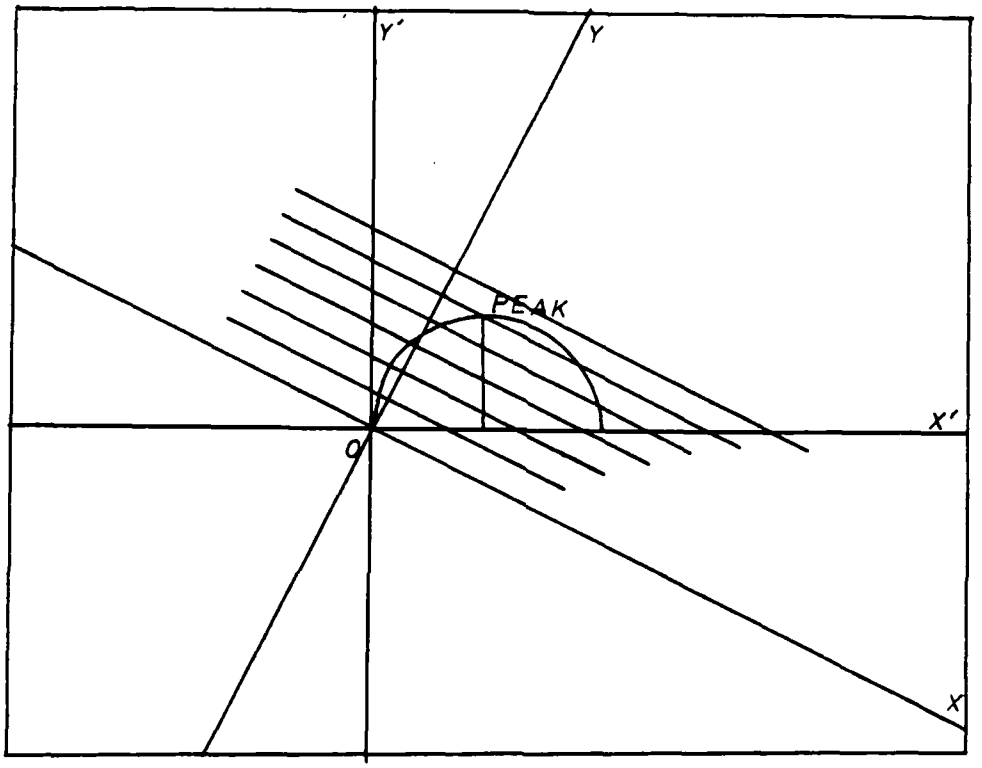


Figure 20. Finding the Interval Boundaries

After the grid intercepts are calculated between the two peaks, they are then ordered by how they proceed along the SINE curve. This means checking their x' coordinate, which always increases along the SINE curve. Each grid intercept is then rotated back to the XY plane.

Summary

This chapter explained the basic mathematics behind the

rotated line drawing algorithm and the algorithm itself. The basic principal of the rotation of axes was explained and how it was expanded to create the rotation algorithm. The rotation algorithm was described and explained, along with its use of the half interval method for the solution of simultaneous transcendental equations.

V. Changes to the Code

There were two major additions to the programs during this thesis effort. The first created a module to perform parallel quantization on the grid intersects. The second major addition provides the ability to rotate the SINE curve and calculate the grid intersects of the rotated drawing.

Parallel Quantization

To implement the parallel quantization scheme, the subroutine GERD, which calculated the triangular quantization scheme was modified. Most of GERD was left intact, but some of the routines that it called were changed, or not used.

PARLEL

The name given to the modified GERD routine is PARLEL, and it is called by the main routine by specifying a new input variable, IPORT. If IPORT equals one, the triangular quantization scheme is used; if it equals two, the parallel quantization scheme is used. The modified code is:

```
404 IF( IPORT .EQ. 1 ) GO TO 405
      CALL PARLEL (JFIN, NSV)
      GO TO 406
405   CALL GERD (JFIN, NSV)
```

CALCND

Instead of calculating the angle between the lines to the nodes and the x-axis, the parallel quantization calculates the slope of the line from the previous node to the

next node, as discussed in Chapter 2. Then, the four points a half of the grid size away from these two nodes are calculated. The two points a half of the grid size away from the next node will lie on the same ring that the next node lies on. Then, each grid intersect on the rings between the current node and the next node are checked to see if they lie between lines drawn between the half grid points. The subroutine CALCND was written to accomplish this. CALCND calculates the points at half of the grid size away the same way the FAR subroutine does. It then checks to see if the slope is zero or infinite, which are the easiest cases to check. If so, then only an upper and lower bound have to be calculated and compared to the point on the next ring. All other point calculations need the slope of the line between the current line and the next node, so the slope is calculated next. The slope is used to calculate the two points which are the intersection of the next inside ring and the two lines a half a grid size away from the current-next node line. The grid intersect on this ring is checked to see whether it is between these two points. If it is, a flag is set; if not, no flag is set and PARLEL will move into the next available ring.

Rotation Algorithm

To implement the rotation algorithm described in Chapter 4, the subroutine ROTATE was written. This subroutine is

called by the main routine when the variable IROTE is set to 2. If IROTE is set to 1, the subroutine FIP is called as was originally programmed. The code to call these routines is as follows:

```
400 IF ( IROTE .EQ. 1 ) GO TO 402
       CALL ROTATE( ALPHA, TSTART, NIPST, NLIMIT, JFIN)
       GO TO 404
402    CALL FIP ( TSTART, NIPST, NLIMIT, JFIN)
```

ROTATE

The ROTATE subroutine finds the first peak of the SINE curve. In the case of the beginning of the curve (where Y is zero), it is really not a 'peak', but it is called that for ease of programming. This 'peak' is then rotated ALPHA radians to its corresponding points in the x'y' plane. Then

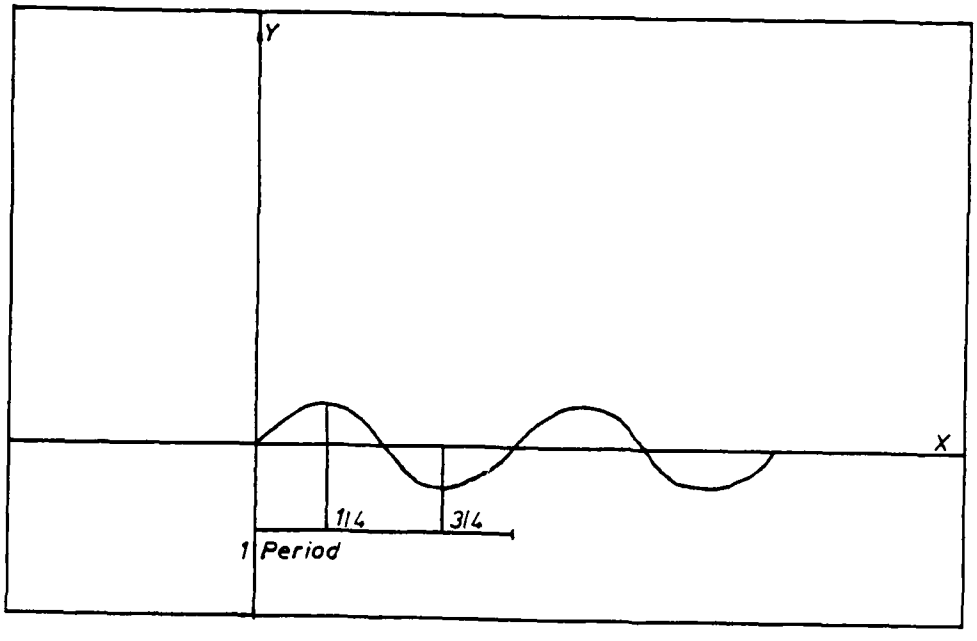


Figure 21. The Peaks of a SINE curve.

the slopes and y-intercept increments of the x' and y' grid lines are found.

After these slopes and y-intercepts are found, the next peak (which is a peak this time) is calculated. The peaks are found by knowing that in the SINE curve, peaks occur at 1/4 and 3/4's of the period as illustrated is Figure 21. This point is also rotated ALPHA radians to find its coordinates in the x'y' plane.

After the two boundary peaks are found, the integer distance between their corresponding x and y coordinates is found. These distances correspond to the number of x' and y' grid lines that are crossed by the curve. The distances are calculated by the function DISTNC.

When these preliminary steps are completed, a check is made to see whether there are any x' grid intersects. If so, the x value of the first boundary peak and the x value of the second boundry peak are sent to the subroutine HALFIN as the two boundary values for the interval. All of the x' grid coordinates are thus calculated.

Now, after the x' grid intercept coordinates are found, there is a check made to see if there are any y' grid intersects in the interval. If there are, the same x boundary values are used to calculate the intercepts of the y' grid coordinates.

These two sets of coordinates are then merged by the path they follow from peak to peak. If there is only one x'

grid intercept or one y' grid intercept, no merging is done and they are just stored. If there are no grid intercepts, the merge section is passed. To merge the two sets of coordinates, the x values of each set are compared to see which is greater. The coordinate with a greater x value is stored and the other is then compared to the next value. Before each value is stored, it is rotated back to the xy plane.

After the x and y grid intercepts are stored from this pass, the next peak value is calculated. The y' intercepts of the x and y grid lines for the new peak distance are calculated, and the program returns to calculating the number of x' and y' grid lines that are crossed. It continues through the algorithm until the end of the SINE curve or the array holding the grid intersects is full.

ROTAXY

This subroutine calculates the new x and y values for a point rotated ALPHA radians from the original point. The data points are checked when they are rotated to their grid line values to see if they are the integer value. If not, they are set to their true integer value.

BXPEAK

This function calculates the y intercept for the first x grid line after a peak. First it takes the y-value of the first of the pair of rotated peaks and finds its integer

value. It then checks to see if the first peak is a high or low peak. If it is a low peak, the y-value is checked to see if it is negative. If the y-value is negative, one is subtracted from the integer value so that when the integer is used, it gives the same relative number to the peak as a positive y-value. The integer value plus one is then multiplied by IX, the distance along the y' axis between the grid line y' intercepts, to give BX, the y' intercept of the next grid line. Now, for a high peak, the difference between the two y-values of the peaks, P1 and P2 is calculated to see if P2 is less than P1. If it is, the method described above is used to calculate BX. If not, the integer value of P1 is multiplied by IX to yield BX.

BYPEAK

This function calculates the y' intercept for the first y grid line after a peak. First it takes the x-value of the first of the pair of rotated peaks and finds its integer value. It then checks to see if the first peak is a high or low peak. If it is a high peak, the x-value is checked to see if it is negative. If the x-value is negative, one is subtracted from the integer value so that when the integer is used, it gives the same relative number to the peak as a positive x-value. The integer value plus one is then multiplied by IY, the distance along the y' axis between the grid line y' intercepts, to give BY, the y' intercept of the next

grid line. Now for a high peak, the difference between the two x-values of the peaks, P1 and P2 is calculated to see if P2 is less than P1. If it is, the method described above is used to calculate By. If not, the integer value of P1 is multiplied by IY to yield BY.

DISTNC

This function calculates the number of x or y grid lines between the two peaks. The X1 and X2 (or Y1 and Y2) values for the two peaks are input along with the grid length, DEL. If the two points are on the opposite side of the y axis (or x axis), the axis must be counted as a grid intercept. The integer values of the two points are subtracted from each other and the absolute value of the difference is found. This absolute value then has one added to it if the axis is crossed and is then divided by DEL. The integer value of the division is sent back as the number of crossed grid lines.

SUMMARY

This chapter discussed the changes made to the code. The first change discussed was the addition of the parallel quantization scheme and how it was implemented. The second change was the implementation of the rotated drawing algorithm from Chapter 4. The discussion of these changes included a summary of all of the subroutines and functions designed to implement them.

VI. Performance Analysis

In this chapter, the performance of the two main changes in the program will be analyzed. First, the parallel quantization scheme will be examined and compared to the previously implemented triangular quantization scheme. Then, the rotated drawing algorithm will be examined amd compared to nonrotated quantizations. The performance of these changes will be analyzed with respect to their area of error and their code rate. Appendix A contains the figures referenced in this chapter which illustrate the code performance.

Parallel Quantization Scheme Performance

The parallel quantization scheme was tested using three different codes; the (1), (1,2), and (1,2,3) codes. For each code set, a set of four periods were used, with four different amplitudes, giving 48 sets of data. The four periods used were 5, 10, 20, and 50; the four amplitudes were 10, 20, 50, and 100. These 48 sets were also run using triangular quantization, so the triangular and the parallel quantization schemes could be compared. The two quantization schemes were compared within the three code types, and any other differences noted.

(1) Code The (1) code performed as expected for the parallel and triangular schemes: they were identical. The area error ranged from .02 for the best amplitude and period to .245 for

the worst. The amplitude of 10 gave the best results, while 100 gave the worst; whereas the period 50 gave the best results and period 5 gave the worst.

It was interesting to note that as the area error rose, so did the bits per unit length (BPL). The BPL ranged from a low of 2.5 to 3.0. The BPL was also the same for the parallel and triangular quantization schemes. It is obvious that for the (1) code in any period and amplitude, the parallel and triangular quantization schemes will give identical results, since the 1-code is independent of the quantization scheme used. The results are a little better for low amplitudes and long periods, which means that if the period is stretched out far enough, the sine curve would almost become a line, but then a code like the (1,2,3) code would be better, which it is as illustrated in Table A-1, because it would have a much smaller BPL than the (1) code.

(1,2) Code In this code an interesting result occurred. While parallel quantization, on the average, was only about .02 above the triangular scheme in the area error, the BPL was quite different. While the BPL for the triangular quantization increased when the amplitude decreased or the period increased (stretched sine curve), the BPL for the parallel quantization decreased for an increase in the amplitude or a decrease in the period. This means that only for a small increment above the area error for the triangular scheme at the low periods and high amplitudes, a BPL up to 1.2 less

than the BPL for the triangular scheme is received. So, the parallel scheme is better for higher amplitudes and lower periods, while the triangular scheme is better for the lower amplitudes and higher periods.

They both start at about the same level for area error, but the parallel is always a little larger. The area error for both schemes ranges from about .05 to .24. The BPL for the parallel starts a little higher than the triangular, when comparing amplitudes, ranging from 2.4 to 3.2, while the triangular ranges from 2.5 to 3.7. The BPL, when compared through periods, also starts a little higher for the parallel, decreasing as the area error decreases. So it would seem that the algorithm could be selected depending upon the aspects of the line drawing to be quantized.

<u>(1,2,3) Code</u> This code performed almost identically to the (1,2) code. The BPL for the parallel scheme is much smaller than the triangular scheme for high amplitudes and low periods, whereas the triangular is better for low amplitudes and high periods. The BPL for the parallel scheme ranges from 2.0 to 3.7, while the triangular ranges from 2.1 to 3.9. Their area error is very close, again within .02, ranging from .05 to .24 for both.

It would seem that, as expected, the (1,2,3) code performs the best with the (1,2) code close behind. Of course, the (1,2,3) code also has a better bit rate than the other two codes. But, between the parallel and the triangular

schemes, the best one would depend upon what kind of line drawing is being encoded; whether it has high amplitudes and low periods or low amplitudes and high periods. If there is a combination, whichever attribute is more important for accurate coding would be the deciding factor for which method is selected.

The Rotated Drawing Algorithm Performance

The rotated drawing algorithm was tested using parallel quantization and the (1), (1,2), and (1,2,3) codes. These jobs were extremely hard to run to completion as some of them take over two hours of processor time to run. Between the high usage of the computer and frequent down time, the jobs would take three to four days to run. It also seems that when the disks were getting too full, the data could not be written and the jobs would abort.

But, of the data that was collected, most of it was really close to the unrotated data for the same curve. In fact, when the job didn't abort, the curve rotated 45 degrees had a smaller area error than the unrotated curve. The only problem is that then it had a greater BPL by about 0.3 to 1.8. It appeared that the curves rotated 5 degrees were at .216 and .484 area error for the amplitudes 10 and 20 respectively. Neither the error nor the BPL changed more than .04 for any runs that were made. This would show a tendancy for the small angles of rotation to be resonably accurate for

flat curves, but not for a curve with a larger curvature. The biggest problem with the rotated data is that it did not give any better results for the (1,2,3) or (1,2) code than the (1) code.

With these outcomes, it would appear that the rotated line drawings do not depend upon what code is used quantize them. They seem to depend more upon what rotation is being used and how the curve looks. This would mean that the quantization of line drawings depends somewhat upon how the grid is placed with respect to the drawing.

Summary

This chapter presented the analysis of the line drawing quantization schemes implemented in this thesis. First, the comparison of the parallel versus the triangular quantization schemes was discussed. Next, the rotated drawing algorithm developed and implemented in this thesis was analyzed. Conclusions were drawn from this analysis and are presented in the next chapter.

VII. Conclusions and Recommendations

This chapter will contain the conclusions drawn from the analysis of the parallel versus triangular quantization schemes and the rotated drawing algorithm. It will also contain any recommendations for future studies in this area.

Conclusions

The analysis of the triangular versus parallel quantization schemes results in two major conclusions. These are:

1) The parallel quantization scheme does better than the triangular quantization scheme when the amplitudes increase and the periods decrease; whereas the triangular quantization does better when the amplitudes decrease and the periods increase.

2) The choice of the parallel versus the triangular quantization schemes will then depend upon whether closer quantization of the high amplitudes or the low amplitudes is desired; also whether closer quantization of the low periods or the high periods is desired.

In the rotated drawing algorithm, it would seem that the algorithm does well. The 45 and 0 degrees give a smaller error rate than the 5 amd 25 degree rotations, and the 45 has an even smaller error rate than no rotation. A better way to run these jobs, would be to split the program apart; using as the first part the selection of the grid intersects, and as

the second part the particular grid quantization scheme. This way the jobs would be much shorter (relatively speaking), and problems would be much easier to locate.

Recommendations for Future Study

The following are recommendations for further study in this area.

1) Continue analysis of the rotated drawing algorithm.

2) Implement the division of the program into two more managable parts.

3) Modify the rotated drawing algorithm so it will analyze the circular function also.

4) Find and implement a better algorithm for finding the rotated grid line intersects than the half interval method.

(From the previous thesis [3:VI-1])

5) Develop a general algorithm capable of approximating the X and Y intersects of any function, whether it is single or multivalued.

6) Study the effect of the position of zero slope of a sinusoid with respect to the horizontal grid lines of the quantizing grid.

7) Quantize a function that does not have a constant peak amplitude, such as the sampling function.

BIBLIOGRAPHY

1. Castor, K. G. and Neuhoff, D. L. "A Rate and Distortion Analysis for Grid Intersect Quantization of Line Drawing Images". Proceedings Eighteenth Annual Allerton Conference on Communication, Control, and Computing, University of Illinois, Urbana-Champaign, October 1980.

2. Castor, K. G. and Neuhoff, D. L. "A Rate and Distortion Analysis for Grid Intersect Encoding of Line Drawings", IEEE 1981 Pattern Recognition & Image Processing Proceedings, IEEE Computer Society Conference on Pattern Recognition and Image Processing, Dallas Texas, August 1981.

3. Christensen, Eric R. "Grid-Based Line Drawing Quantization". Master Thesis AFIT/GE/EE/83D-16. Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio. December 1983.

4. Freeman, Herbert. "Computer Processing of Line Drawing Images", Computing Surveys, 6, 1: 57-97 (March 1974).

5. Freeman, Herbert. "Application of the Generalized Chain Coding Scheme to Map Data Processing", Proceedings IEEE Computer Society Conference on Pattern Recognition and Image Processing. Chicago, Illinois. May 1978.

6. Freeman, Herbert and Saghri, J. A. "Generalized Chain Codes for Planar Curves", Proceedings of the 4th International Joint Conference on Pattern Recognition. Kyoto, Japan. November 1978.

7. Freeman, Herbert. "Efficient Representation of Spatial Data", IEEE 1981 Pattern Recognition & Image Processing Proceedings, IEEE Computer Society Conference on Pattern Recognition and Image Processing. Dallas Texas. August 1981.

8. Freeman, Herbert and Frauenknecht, P. J. "An Efficient Computer Representation Scheme for Linear Map Data", IEEE 1982 Pattern Recognition & Image Processing Proceedings.

9. Glass, Jeremy. "A Criterion for the Quantization of Line Drawing Data" Doctoral Disseration, New York University, June 1965.

10. Jones, Keith. "Grid-Based Line Drawing Quantization". Master Thesis AFIT/GE/EE/82D-41. Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio. December 1982.

11. Jurgensen, R. C., Donnelly, A. J., Dolciani, M. P. Modern School Mathematics - Geometry, Houghton Mifflin Co., Boston, 1969.

12. Kindle, Joseph H. "Theory and Problems of Plane and Solid Analytic Geometry", Shaum's Outline Series, McGraw - Hill Book Co., 1950.

13. Koplowitz, Jack. "On the Performance of Chain Codes for Quantization of Line Drawings", IEEE Transactions on Pattern Analysis and Machine Intelligence, 3, 2: 180-185 (March 1981).

14. Kuo, Shan S. Numerical Methods and Computers, Addison - Wesley, Reading, Mass., 1965.

15. Maxwell, P. C. "The Perception and Description of Line Drawings by Computer", Computer Graphics and Image Processing, 1: 31-46 (April 1972).

16. Saghri, John A. Efficient Encoding of Line Drawing Data with Generalized Chain Codes, Tech Rept. IPL-TR-79-003, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, N.Y., August 1979.

Appendix A

Performance Plots

In each figure, the upper plot is of the accumulated encoding rate per unit length plotted against the domain of the function. The lower plot is of the accumulated area error per unit length plotted against the domain of the function. Each figure is noted with the specific parameters to create the lines, and the variable parameter values for the parameter that varies on that graph.

LIST OF FIGURES

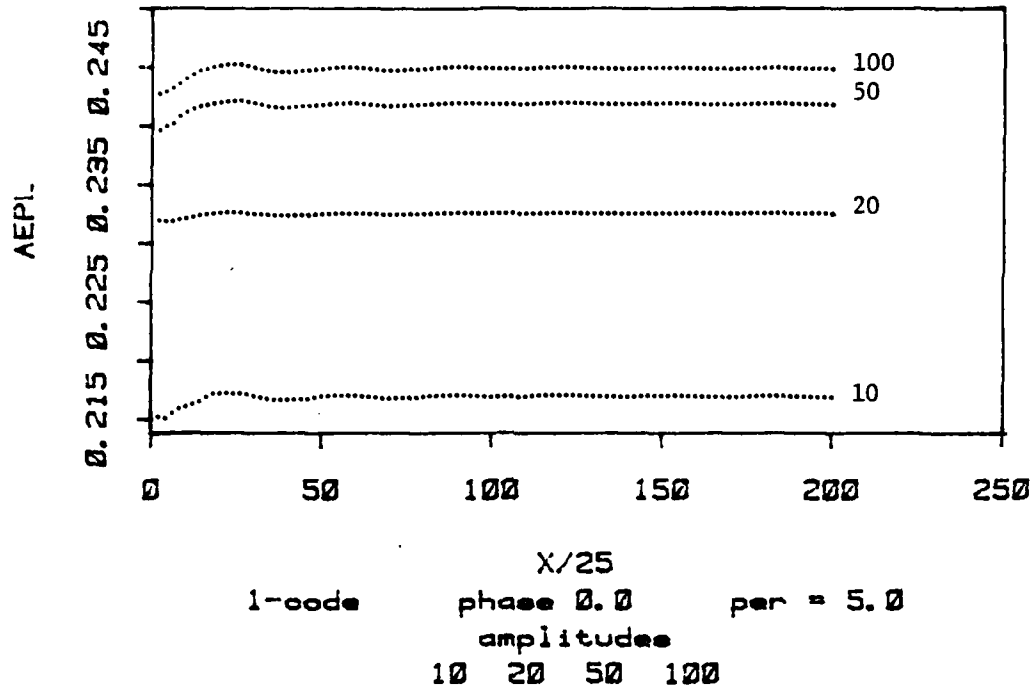
Figure		Page
A-1	(1) Code, Per = 5, A = 10 - 100, Parallel	63
A-2	(1) Code, Per = 5, A = 10 - 100, Triangular . . .	64
A-3	(1,2) Code, Per = 5, A = 10 - 100, Parallel . . .	65
A-4	(1,2) Code, Per = 5, A = 10 - 100, Triangular . .	66
A-5	(1,2,3) Code, Per = 5, A = 10 - 100, Parallel . .	67
A-6	(1,2,3) Code, Per = 5, A = 10 - 100, Triangular .	68
A-7	(1) Code, Per = 10, A = 10 - 100, Parallel	69
A-8	(1) Code, Per = 10, A = 10 - 100, Triangular . . .	70
A-9	(1,2) Code, Per = 10, A = 10 - 100, Parallel . . .	71
A-10	(1,2) Code, Per = 10, A = 10 - 100, Triangular . .	72
A-11	(1,2,3) Code, Per = 10, A = 10 - 100, Parallel . .	73
A-12	(1,2,3) Code, Per = 10, A = 10 - 100, Triangular .	74
A-13	(1) Code, Per = 20, A = 10 - 100, Parallel	75
A-14	(1) Code, Per = 20, A = 10 - 100, Triangular . . .	76
A-15	(1,2) Code, Per = 20, A = 10 - 100, Parallel . . .	77
A-16	(1,2) Code, Per = 20, A = 10 - 100, Triangular . .	78
A-17	(1,2,3) Code, Per = 20, A = 10 - 100, Parallel . .	79
A-18	(1,2,3) Code, Per = 20, A = 10 - 100, Triangular .	80
A-19	(1) Code, Per = 50, A = 10 - 100, Parallel	81
A-20	(1) Code, Per = 50, A = 10 - 100, Triangular . . .	82
A-21	(1,2) Code, Per = 50, A = 10 - 100, Parallel . . .	83
A-22	(1,2) Code, Per = 50, A = 10 - 100, Triangular . .	84
A-23	(1,2,3) Code, Per = 50, A = 10 - 100, Parallel . .	85

A-24 (1,2,3) Code, Per = 50, A = 10 - 100, Triangular . 86
A-25 (1) Code, Per = 5, Rotat = 0 - 45, A = 10 87
A-26 (1) Code, Per = 5, Rotat = 0 - 45, A = 20 88
A-27 (1,2) Code, Per = 5, Rotat = 0 - 45, A = 10 . . . 89
A-28 (1,2) Code, Per = 5, Rotat = 0 - 45, A = 20 . . . 90
A-29 (1,2,3) Code, Per = 5, Rotat = 0 - 45, A = 10 . . 91
A-30 (1,2,3) Code, Per = 5, Rotat = 0 - 45, A = 20 . . 92
A-31 (1) Code, Per = 10, Rotat = 0 - 45, A = 10 93
A-32 (1) Code, Per = 10, Rotat = 0 - 45, A = 20 94
A-33 (1,2) Code, Per = 10, Rotat = 0 - 45, A = 10 . . . 95
A-34 (1,2) Code, Per = 10, Rotat = 0 - 45, A = 20 . . . 96
A-35 (1,2,3) Code, Per = 10, Rotat = 0 - 45, A = 10 . . 97
A-36 (1,2,3) Code, Per = 10, Rotat = 0 - 45, A = 20 . . 98
A-37 (1,2) Code, Per = 20, Rotat = 0 - 45, A = 10 . . . 99
A-38 (1,2,3) Code, Per = 20, Rotat = 0 - 45, A = 10 . . 100

APPENDIX A

LIST OF TABLES

Table		Page
A-1	Asymptotic Values for AEPL and BPL for All Codes All Periods, and All Amplitudes Seperate Columns for Parallel and Triangular Quantization	101
A-2	Asymptotic Values for AEPL and BPL for All Codes All Periods, Amplitudes = 10, 20	103


AEPL
0.215 0.225 0.235 0.245
100
50
20
10
0
50
100
150
200
250
X/25
1-code phase 0.0 per = 5.0
amplitudes
10 20 50 100

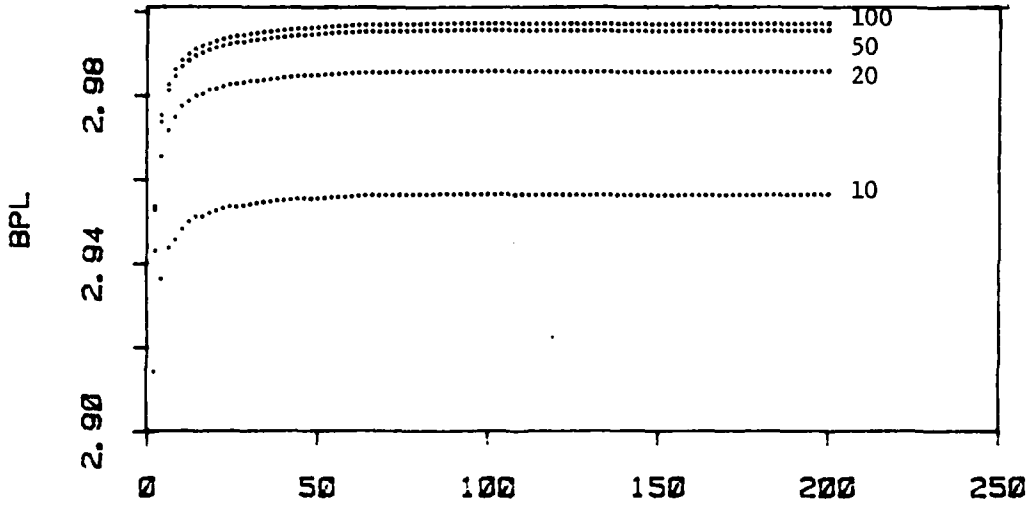
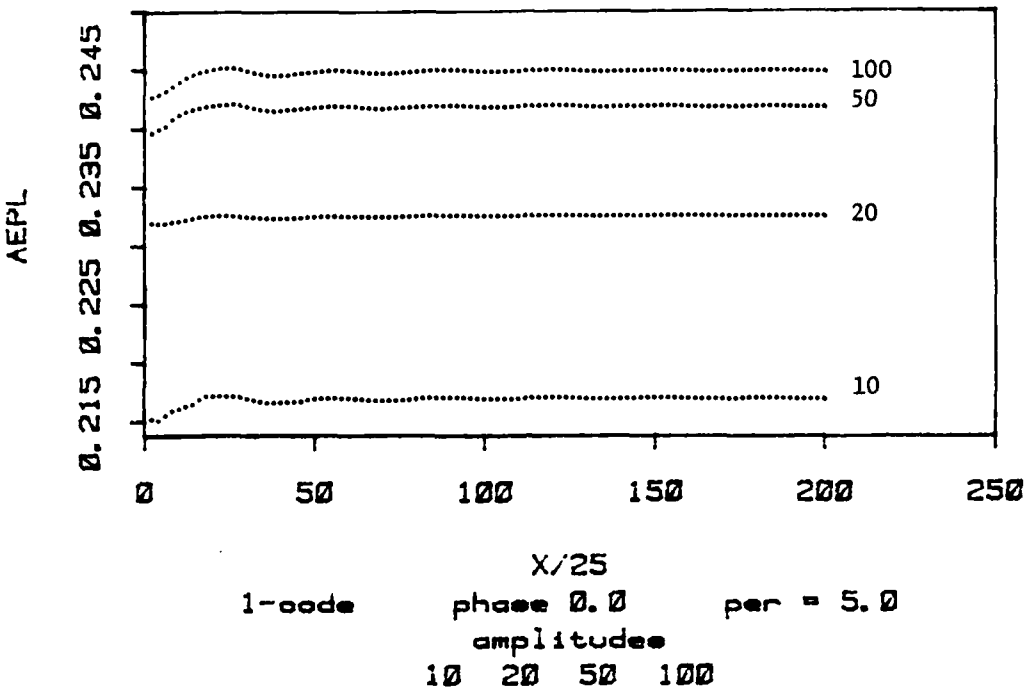
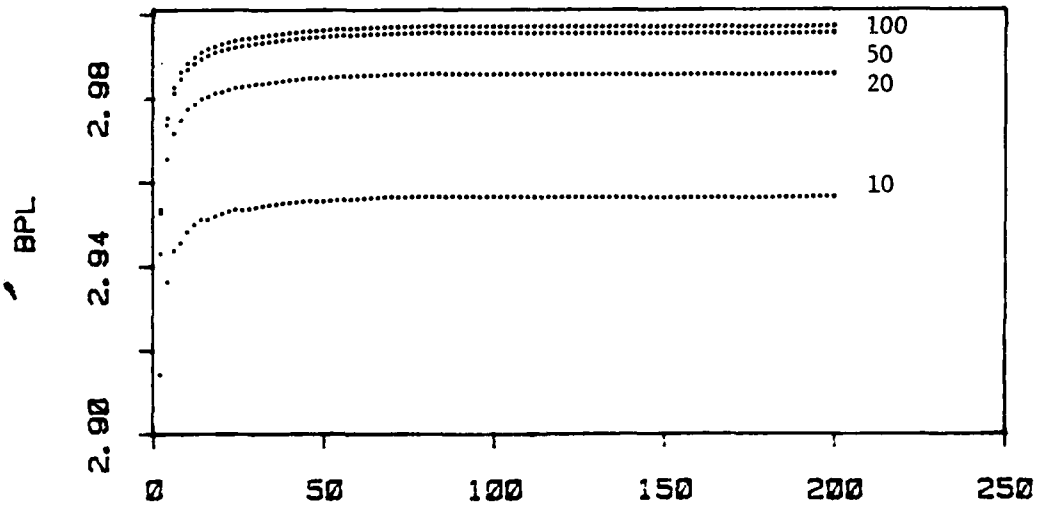

BPL
2.90 2.94 2.98
100
50
20
10
0
50
100
150
200
250
X/25
Parallel Quantization

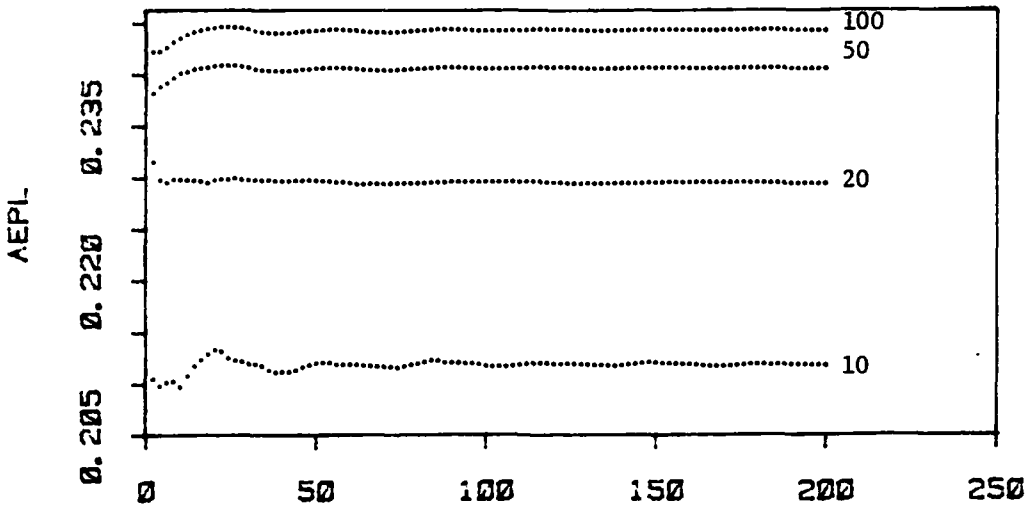
Figure A-1





Triangular Quantization

Figure A-2

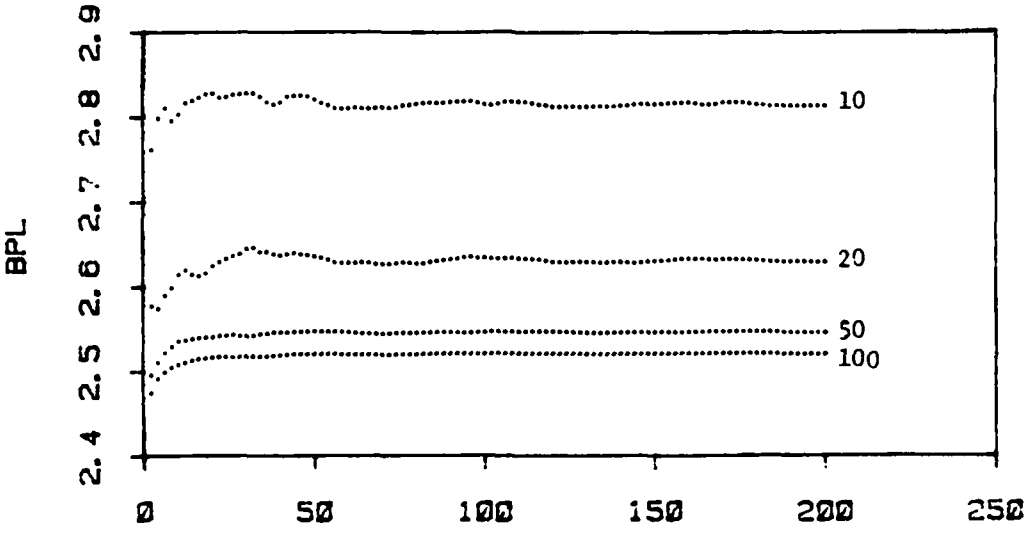


X/25

12-code phase 0.0 per = 5.0

amplitudes

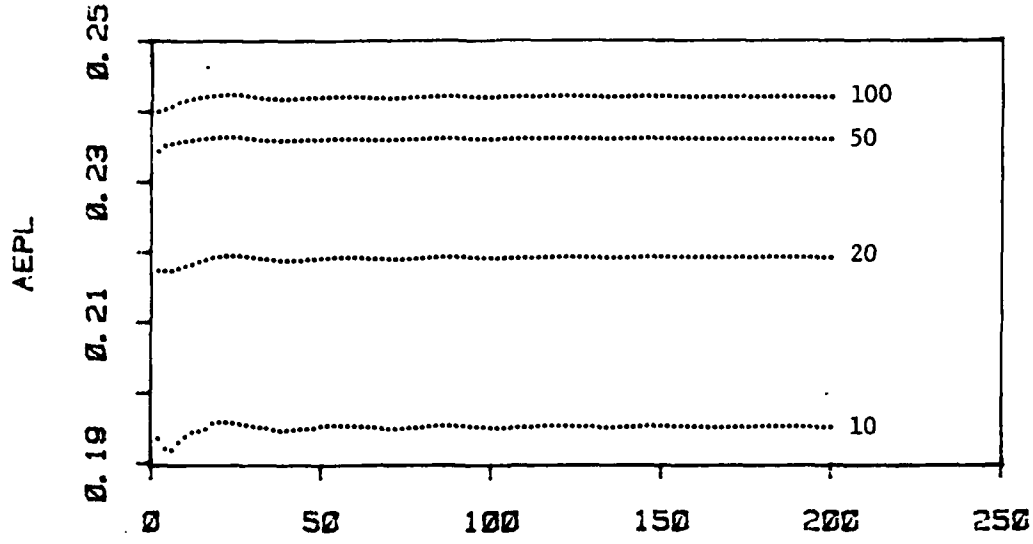
10 20 50 100



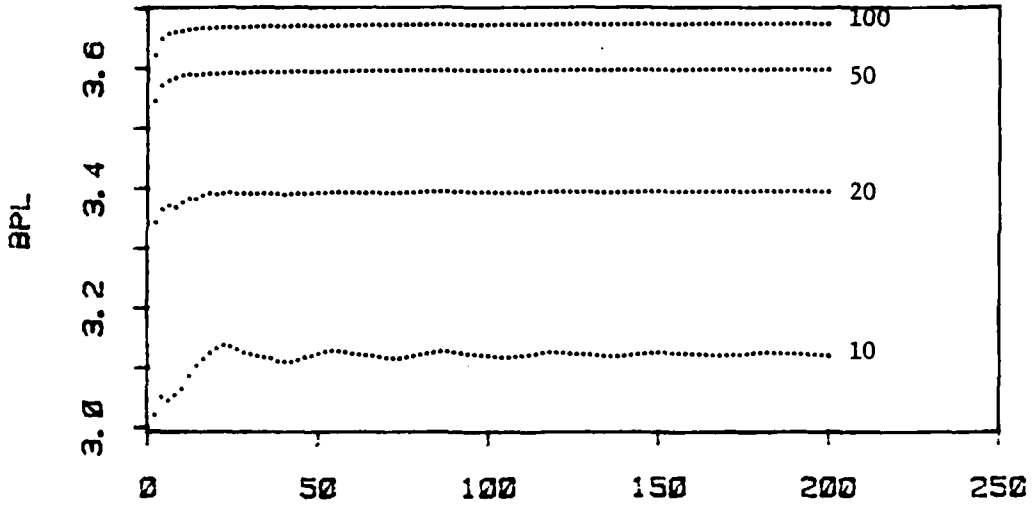
X/25

Parallel Quantization

Figure A-3

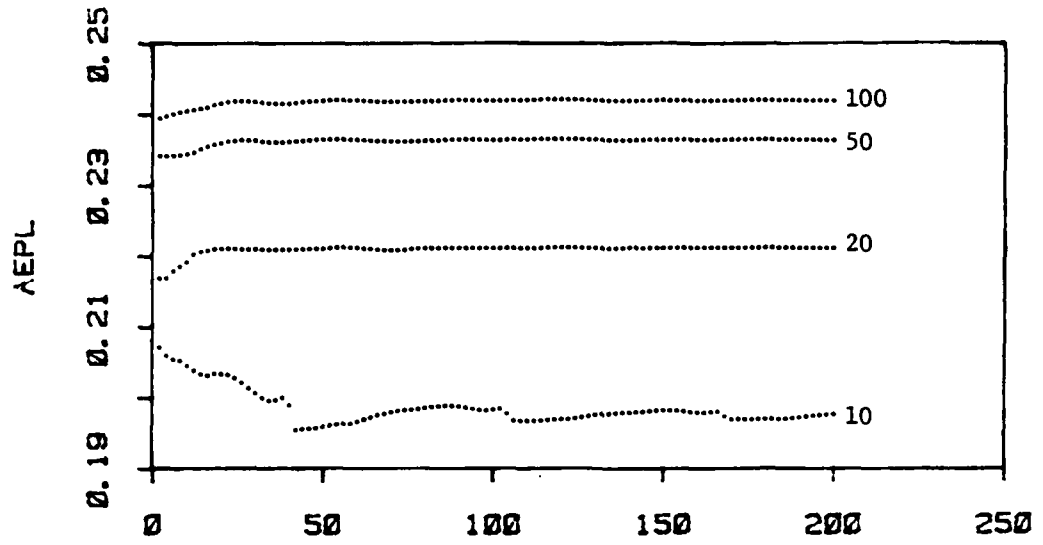


12-code phase 0.0 per = 5.0
amplitudes
10 20 50 100



Triangular Quantization

Figure A-4

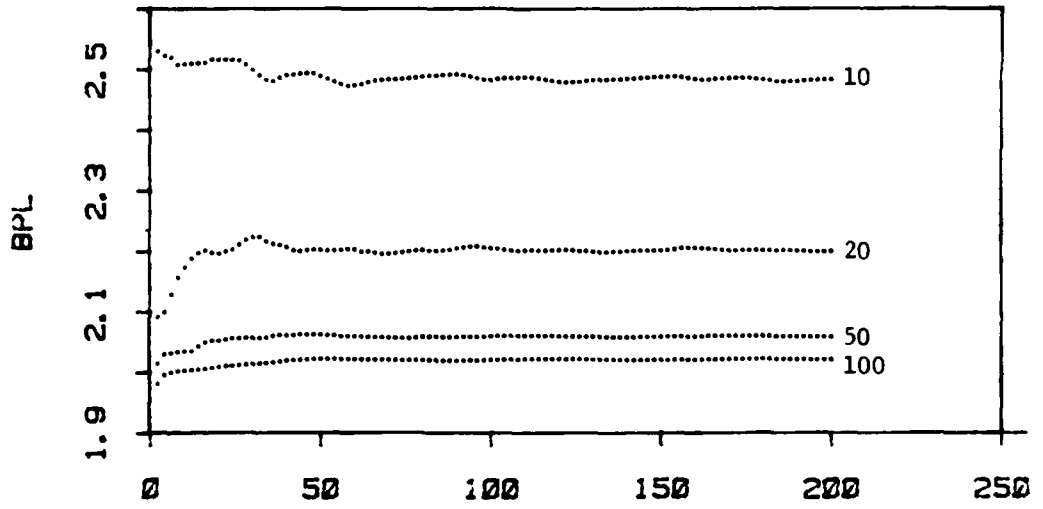


X/25

123-code phase 0.0 per = 5.0

amplitudes

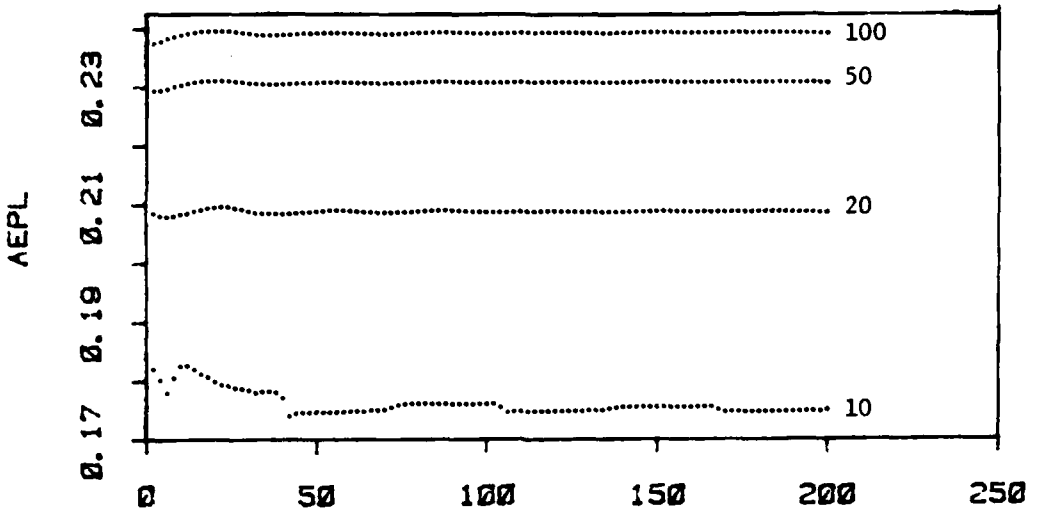
10 20 50 100



X/25

Parallel Quantization

Figure A-5

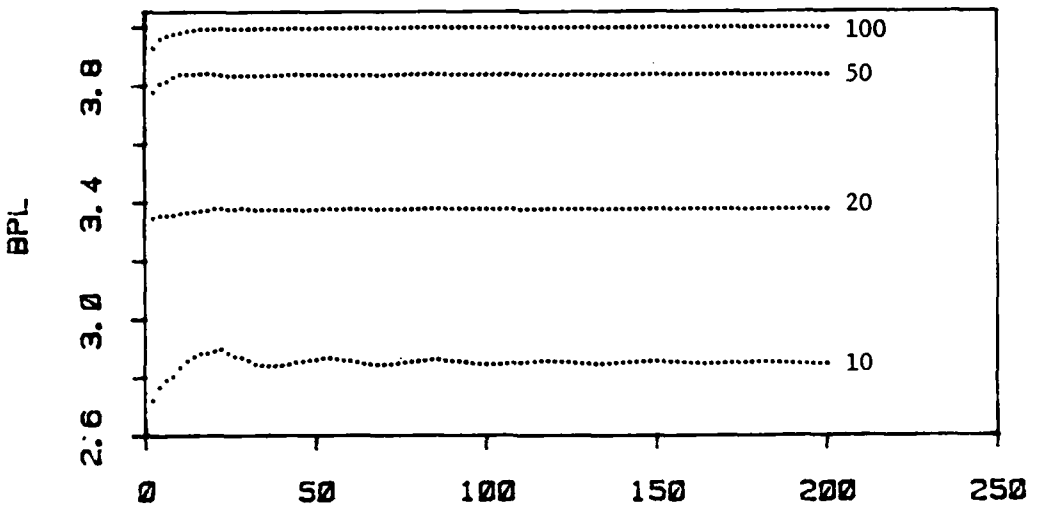


X/25

123-code phase 0.0 per = 5.0

amplitudes

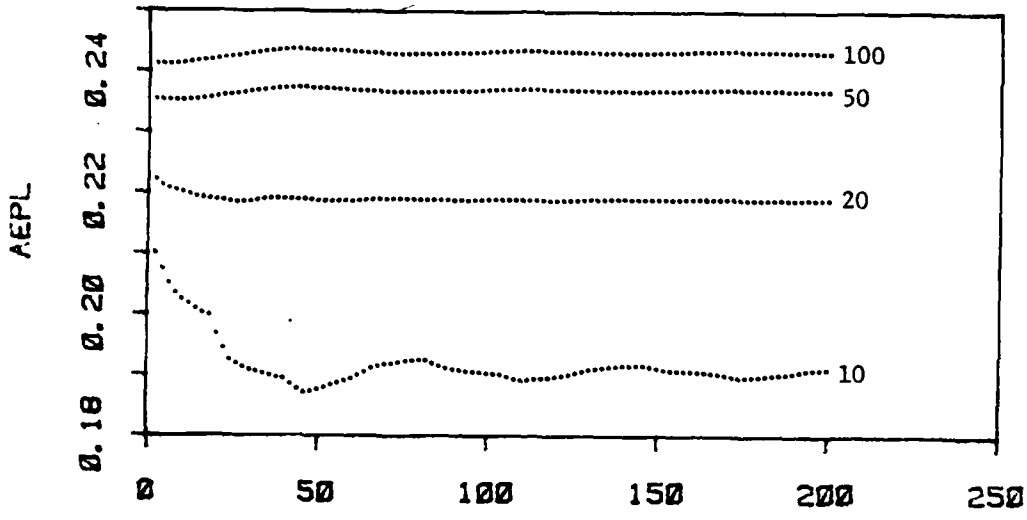
10 20 50 100



X/25

Triangular Quantization

Figure A-6


AEPL
0.18 0.20 0.22 0.24
0 50 100 150 200 250
100
50
20
10
X/50
i-code phase 0.0 per = 10.0
amplitudes
10 20 50 100

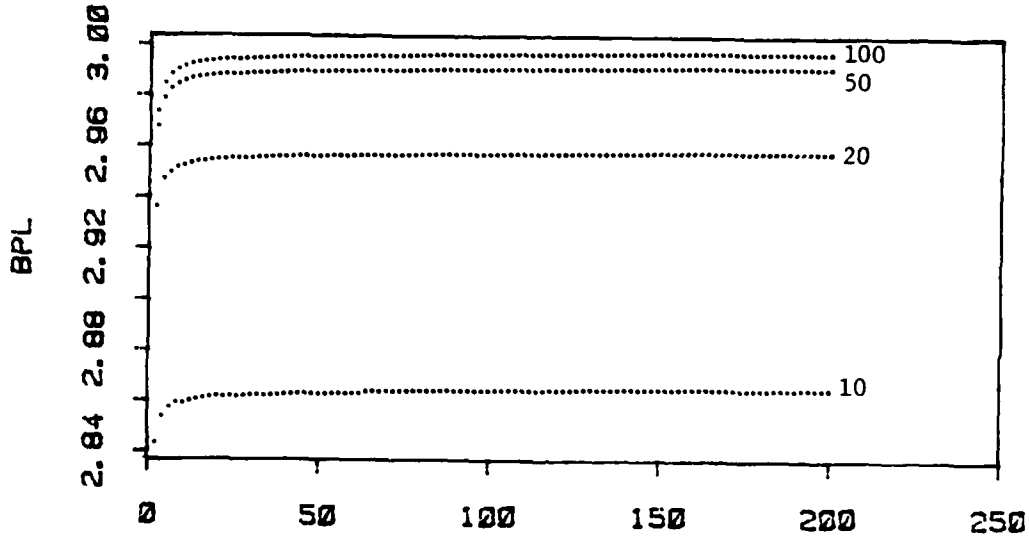
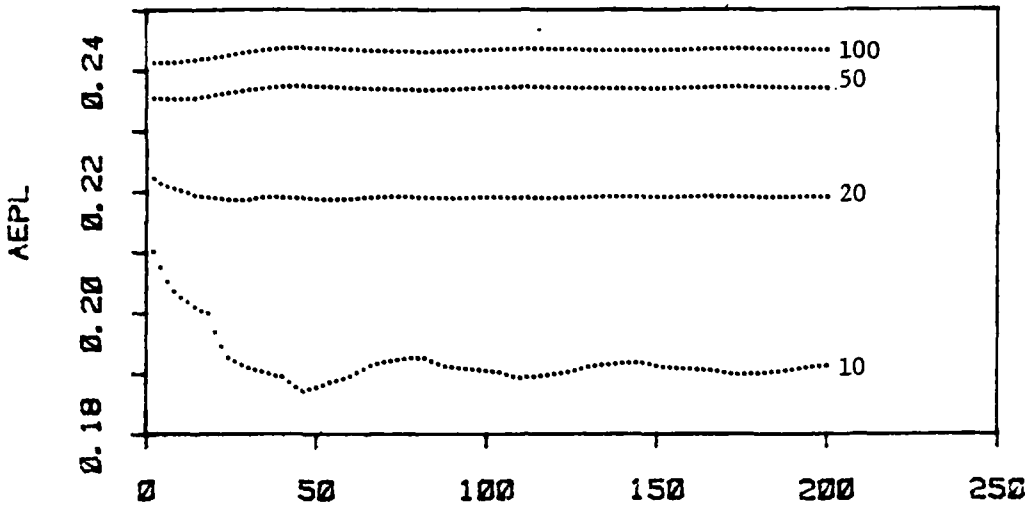

BPL
2.84 2.88 2.92 2.96 3.00
0 50 100 150 200 250
100
50
20
10
X/50
Parallel Quantization

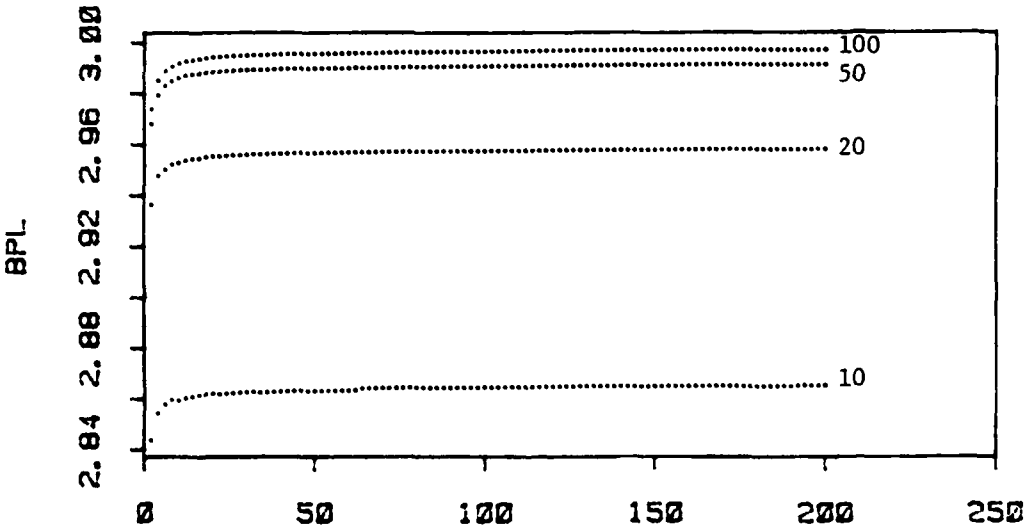
Figure A-7



1-code phase 0.0 per = 10.0

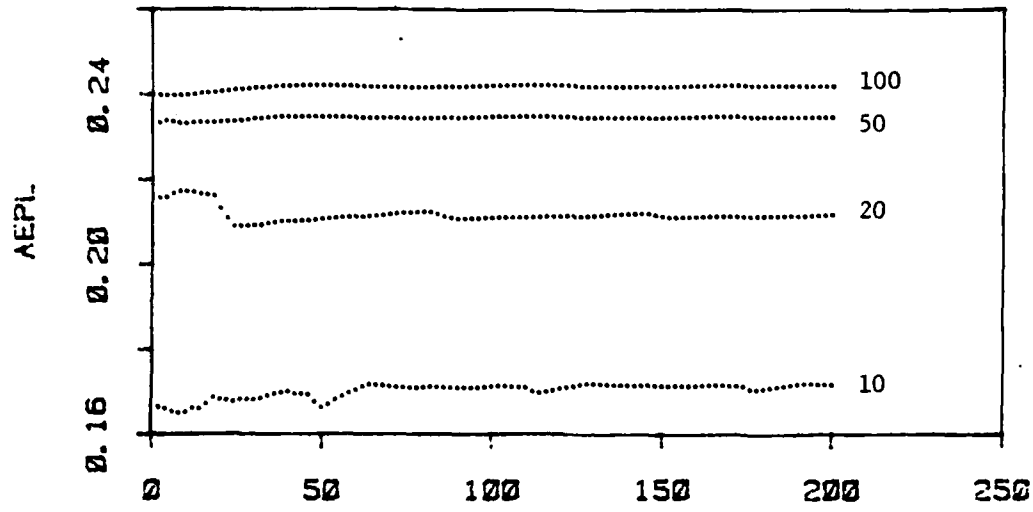
amplitudes

10 20 50 100



Triangular Quantization

Figure A-8


AEPL
0.24
0.20
0.16
100
50
20
10
0
50
100
150
200
250
X/50
12-code phase 0.0 per = 10.0
amplitudes
10 20 50 100

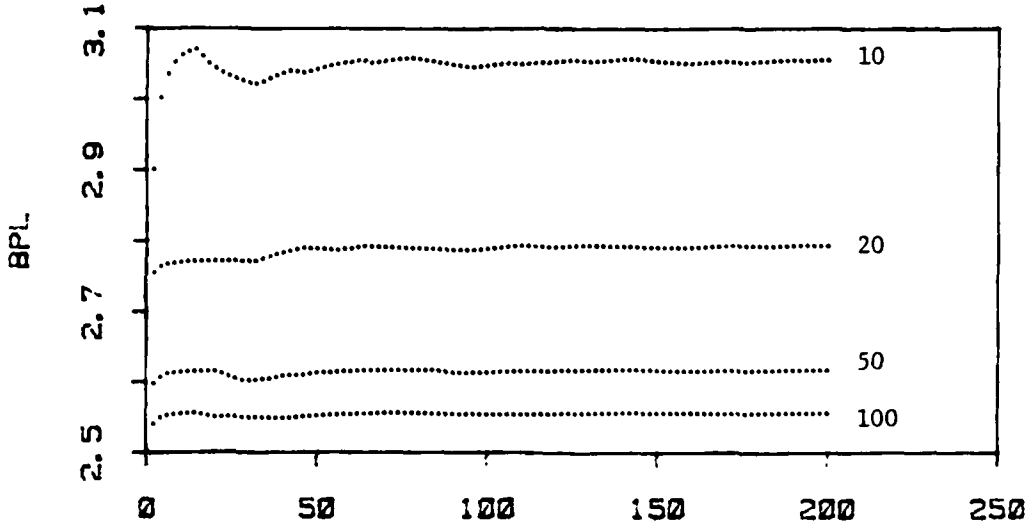
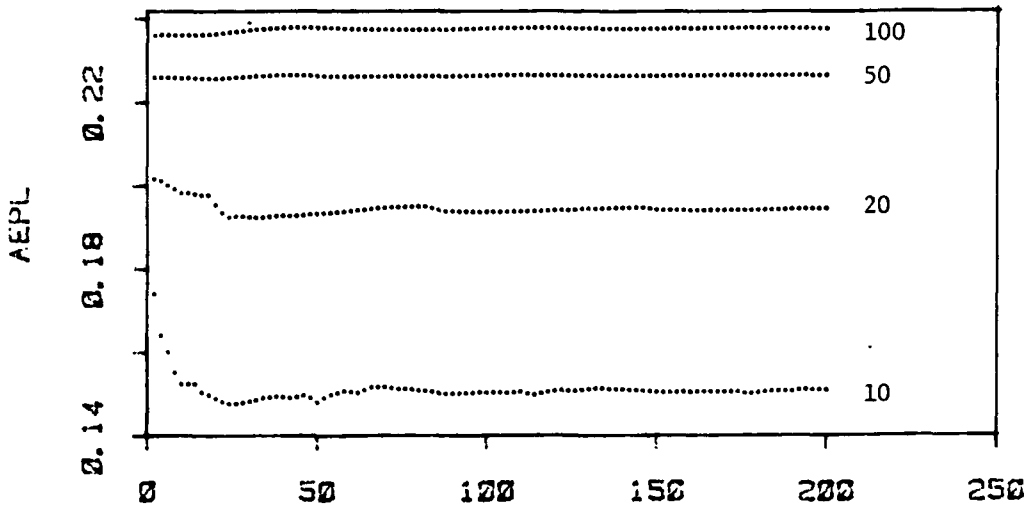

BPL
3.1
2.9
2.7
2.5
10
20
50
100
0
50
100
150
200
250
X/50
Parallel Quantization

Figure A-9


AEPL
0.22
0.18
0.14
100
50
20
10
0
50
100
150
200
250
X/50
12-code phase 0.0 per = 10.0
amplitudes
10 20 50 100

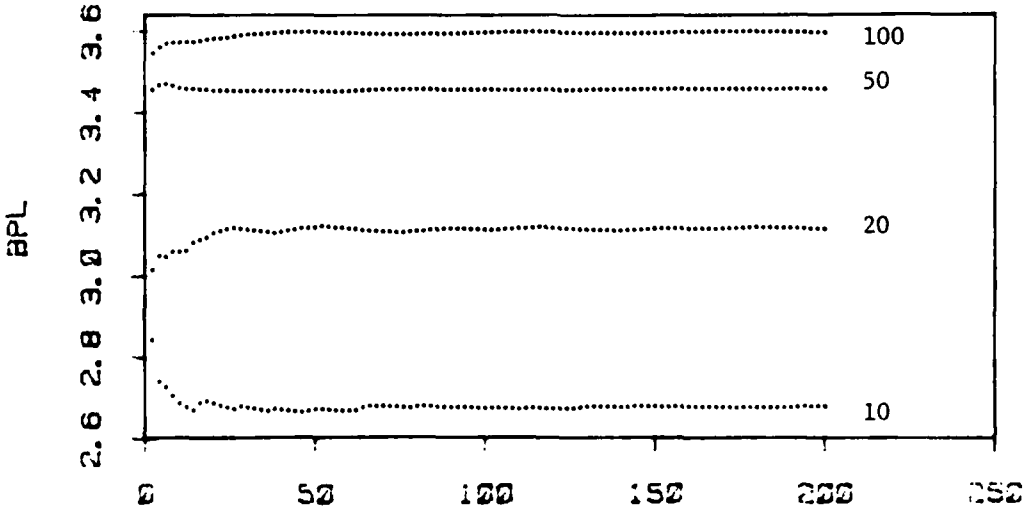
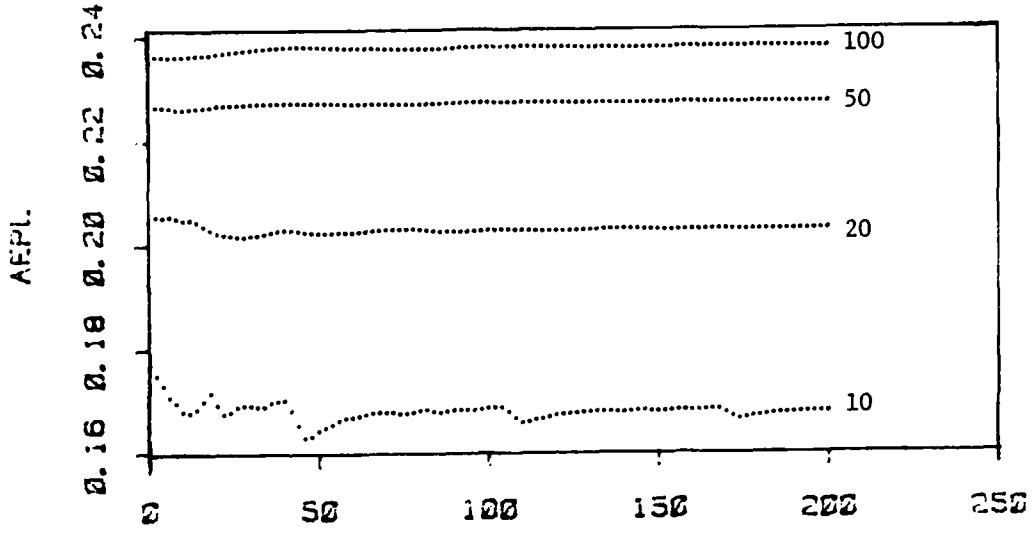

BPL
3.6
3.4
3.2
3.0
2.8
2.6
100
50
20
10
0
50
100
150
200
250
X/50
Triangular Quantization

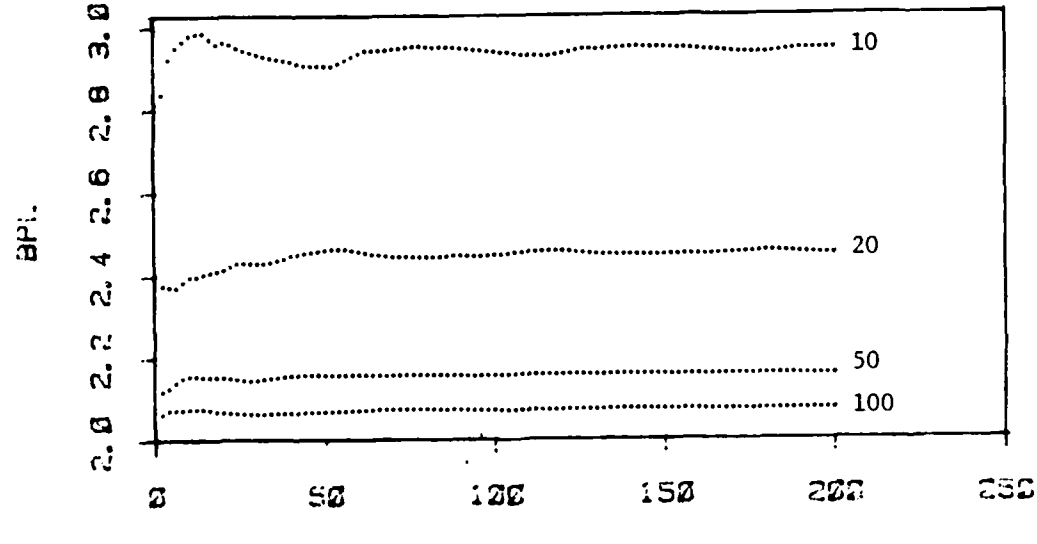
Figure A-10



123-code phase 0.0 per = 10.0

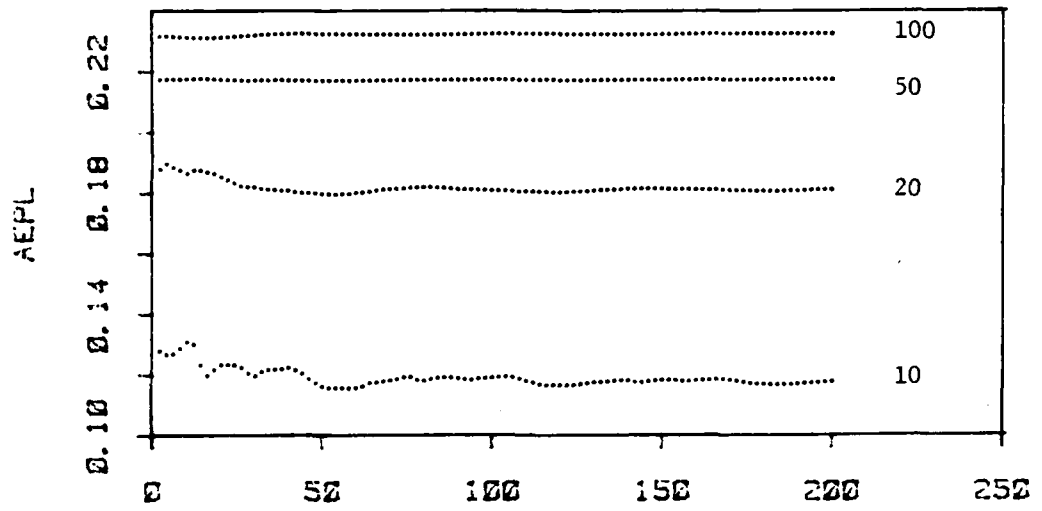
amplitudes

10 20 50 100



Parallel Quantization

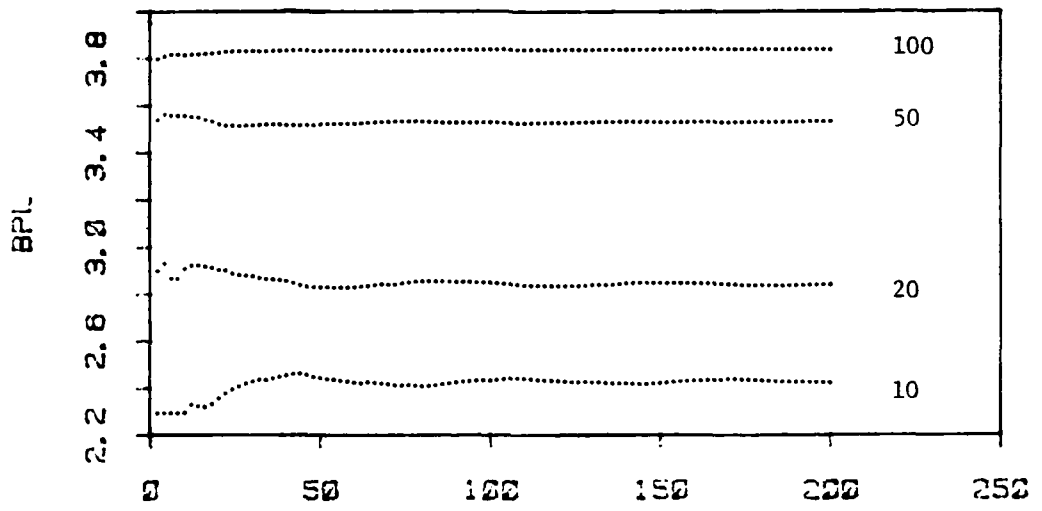
Figure A-11



123-code phase 0.0 per = 10.0

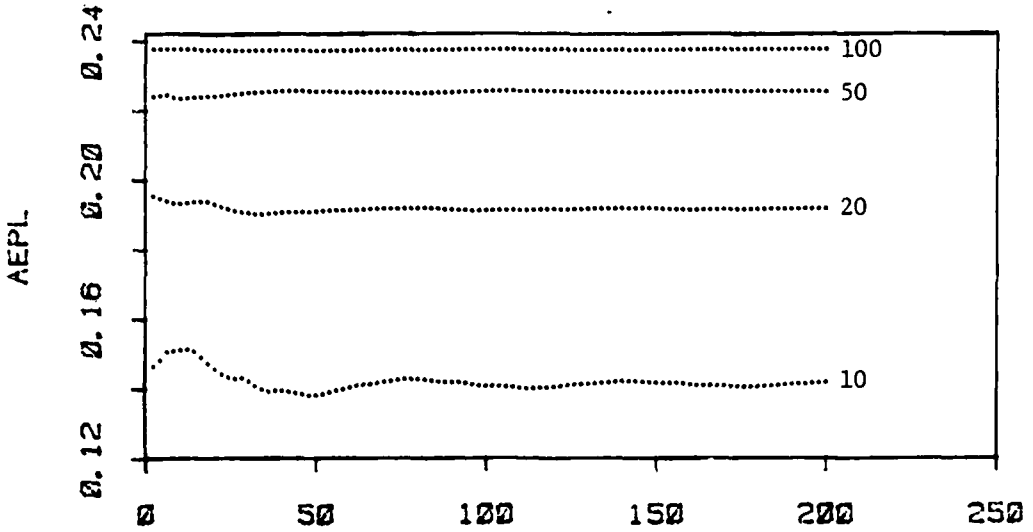
amplitudes

10 20 50 100



Triangular Quantization

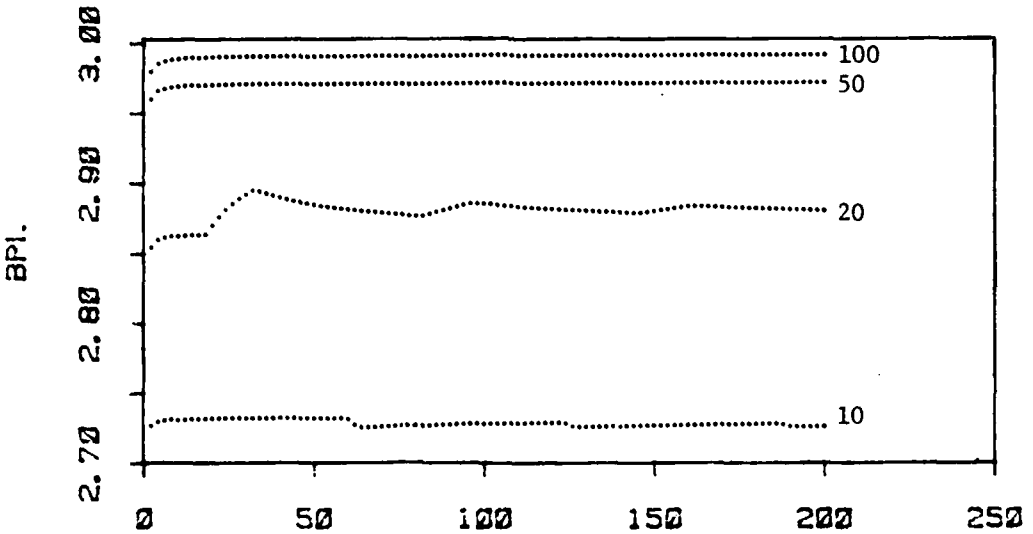
Figure A-12



1-code phase 0.0 per = 20.0

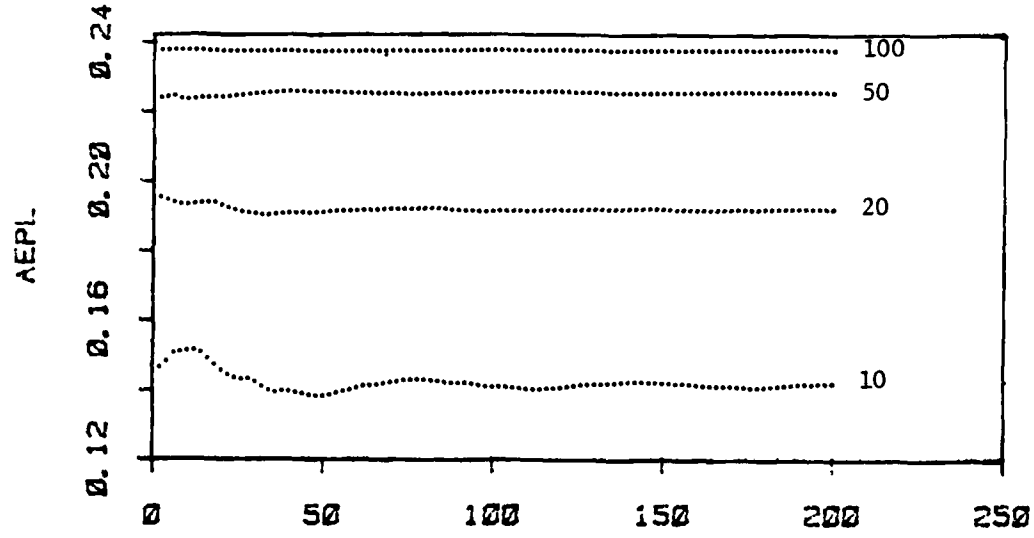
amplitudes

10 20 50 100

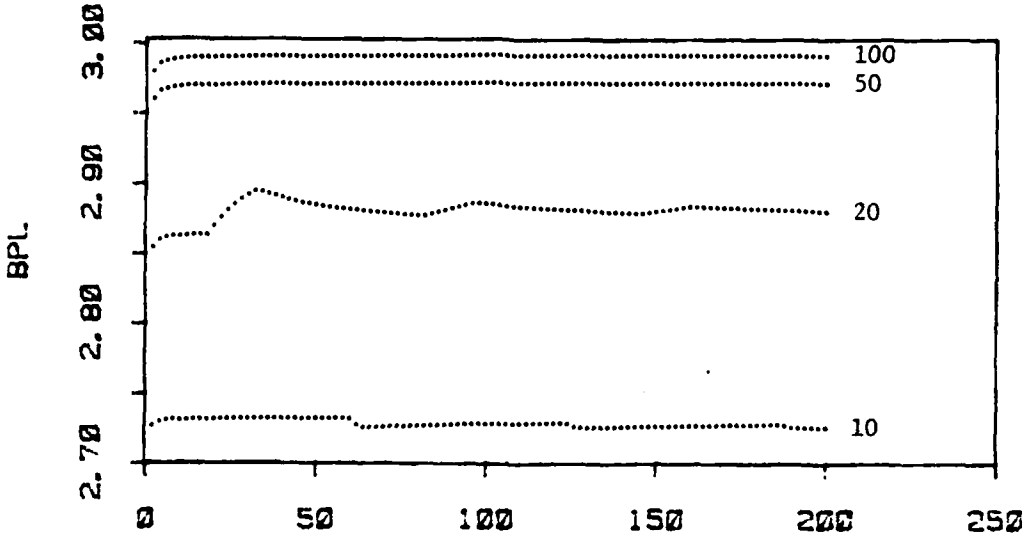


Parallel Quantization

Figure A-13

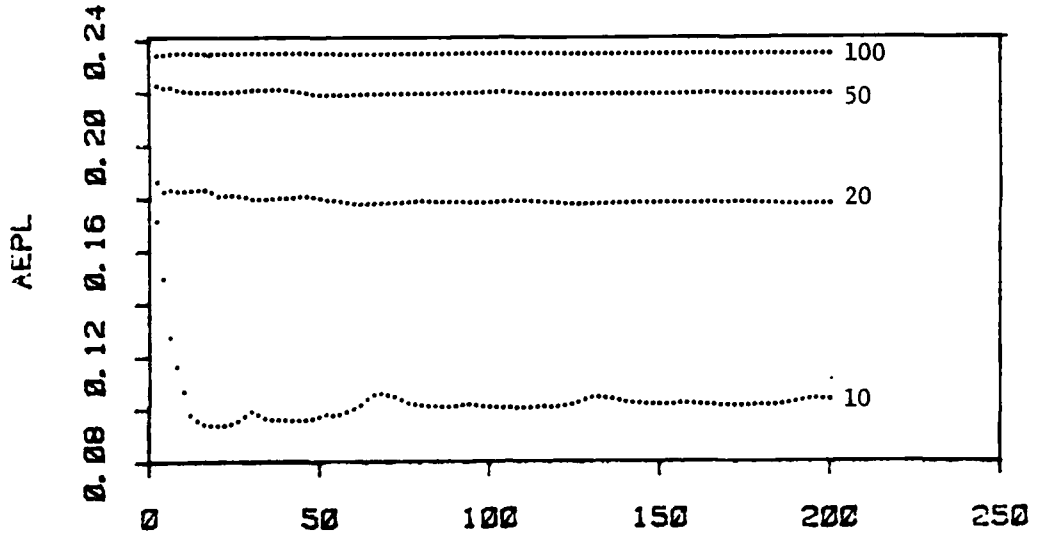


1-code phase 0.0 per = 20.0
amplitudes
10 20 50 100



Triangular Quantization

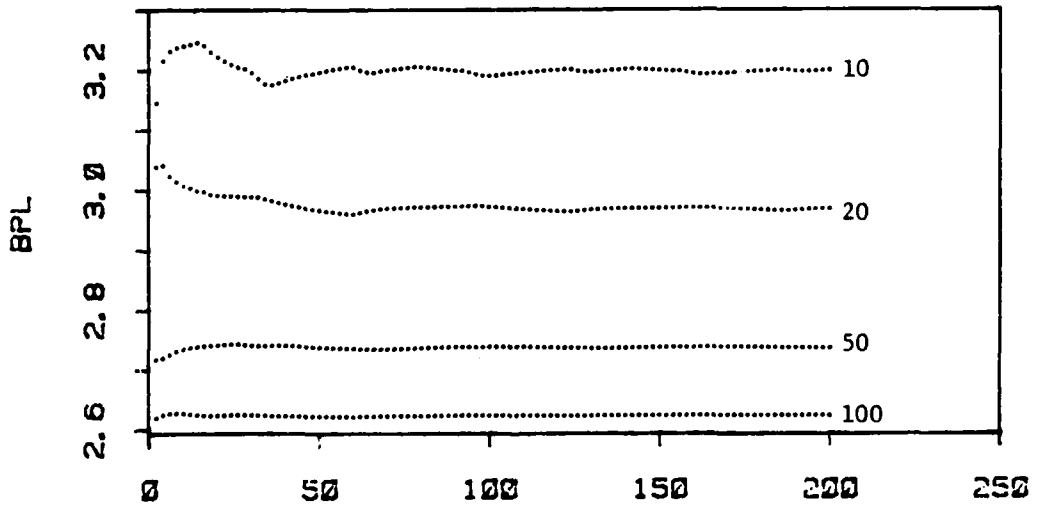
Figure A-14



12-code phase 0.0 per = 20.0

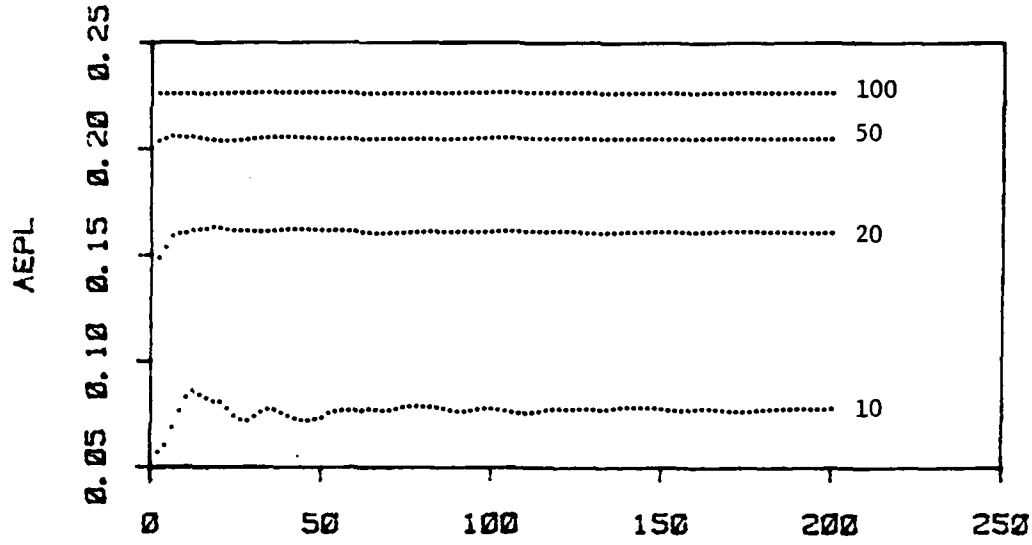
amplitudes

10 20 50 100

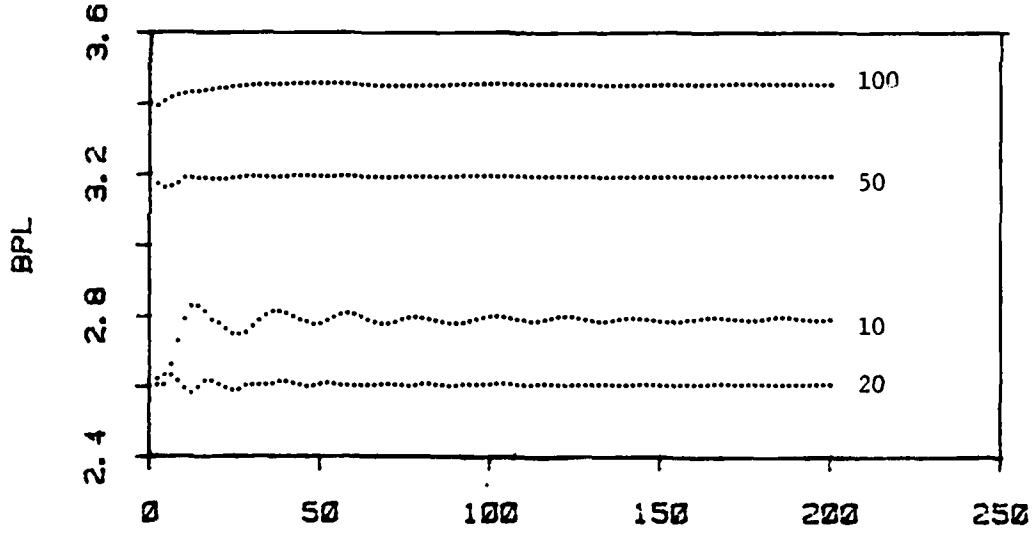


Parallel Quantization

Figure A-15

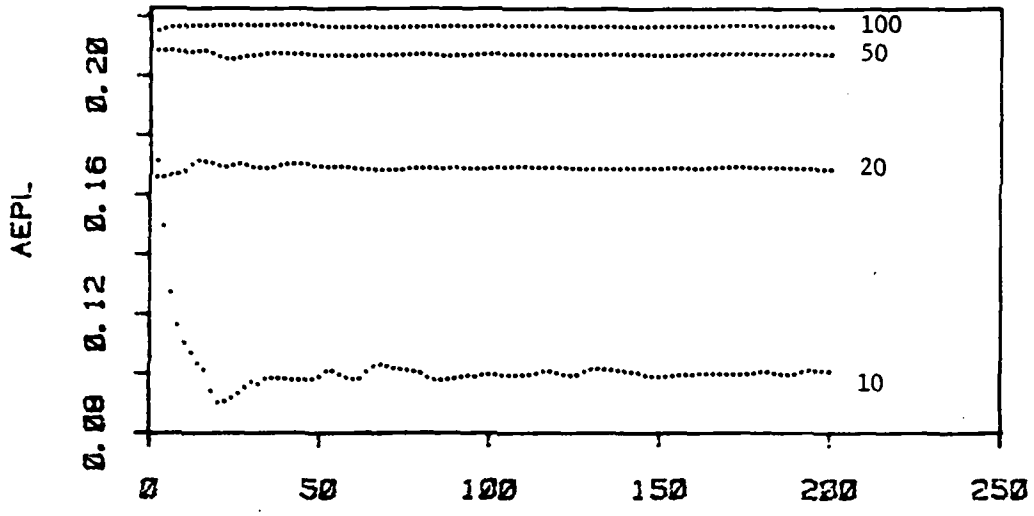


12-code phase 0.0 per = 20.0
amplitudes
10 20 50 100



Triangular Quantization

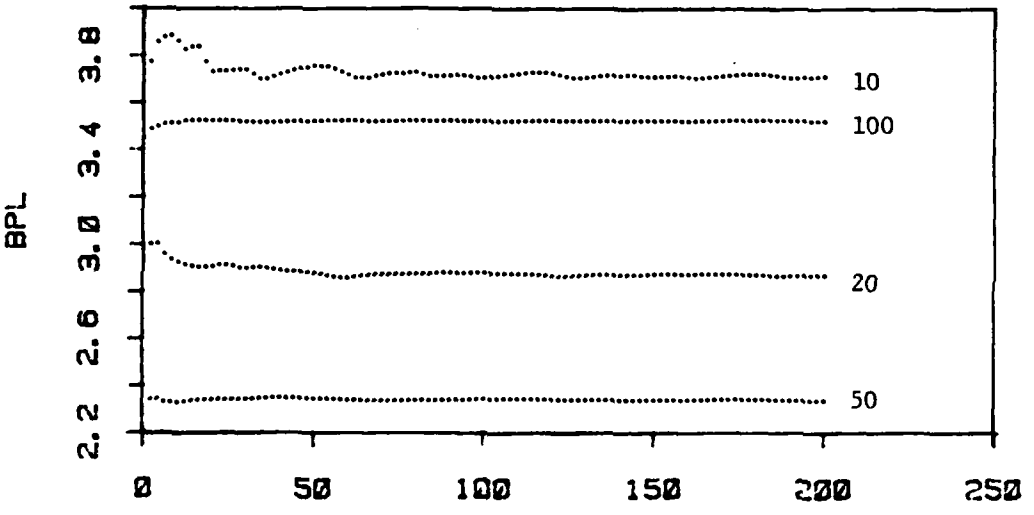
Figure A-16



123-code phase 0.0 per = 20.0

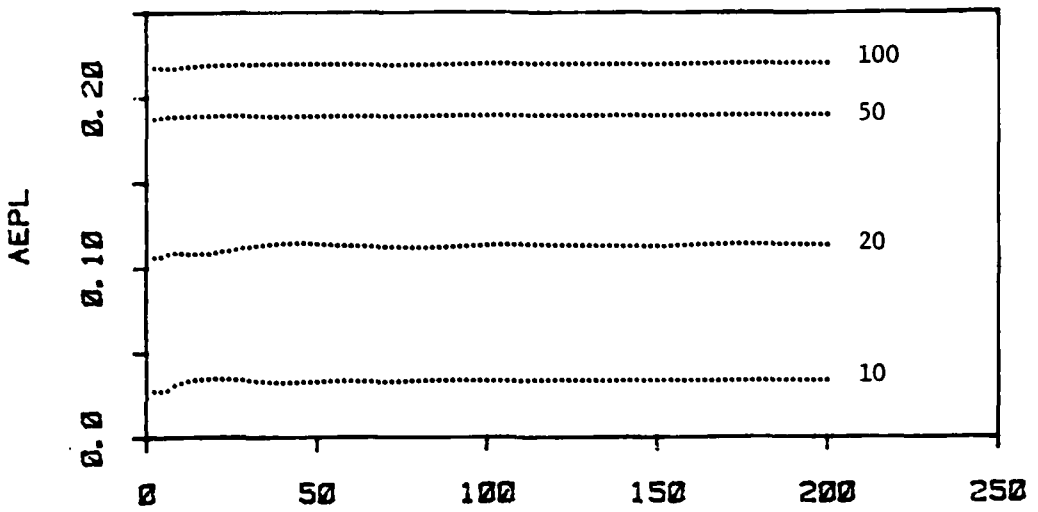
amplitudes

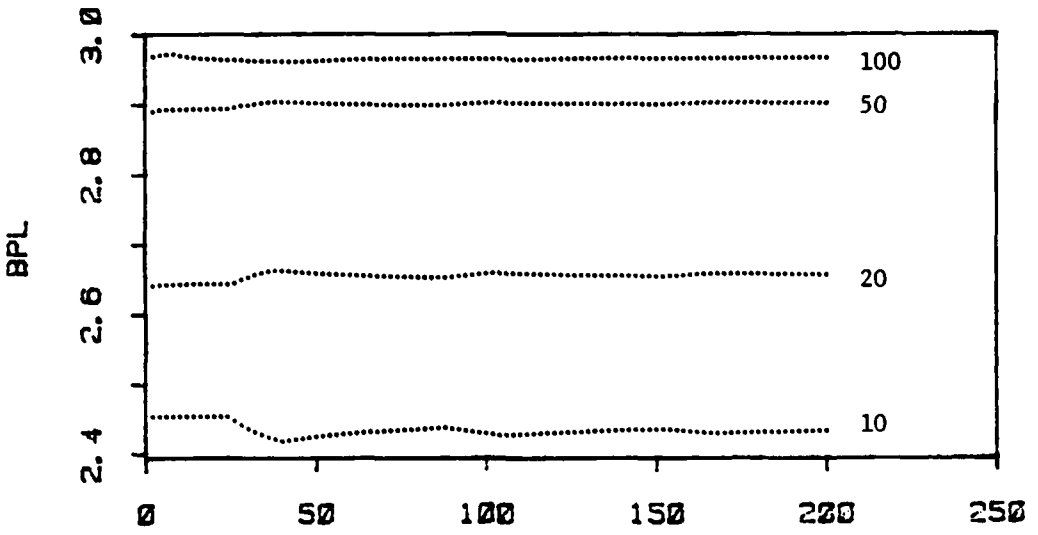
10 20 50 100



Parallel Quantization

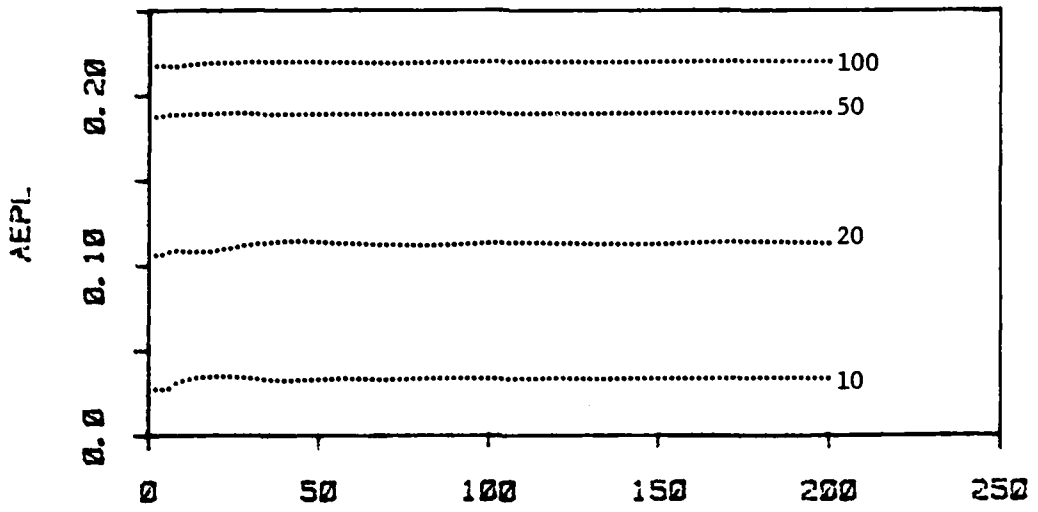
Figure A-17



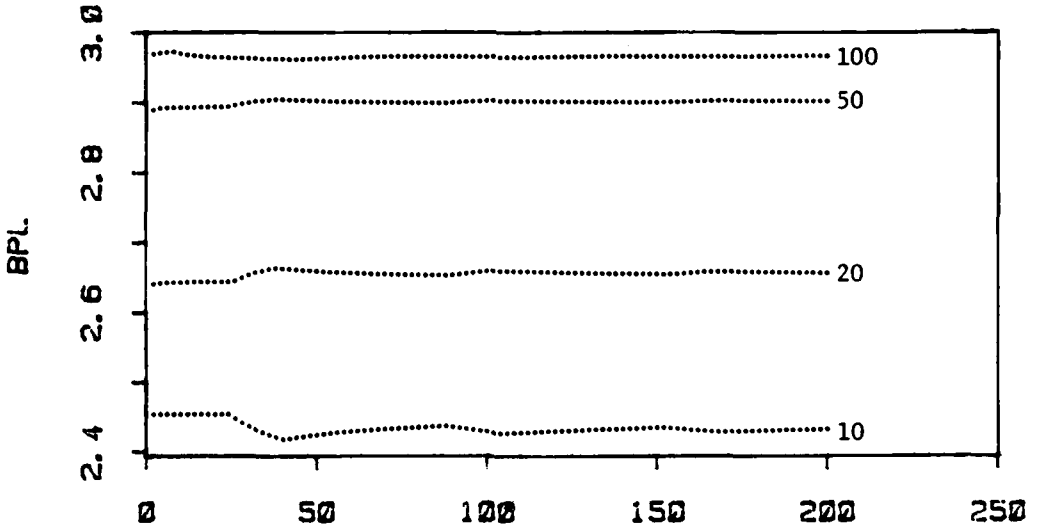


Parallel Quantization

Figure A-19

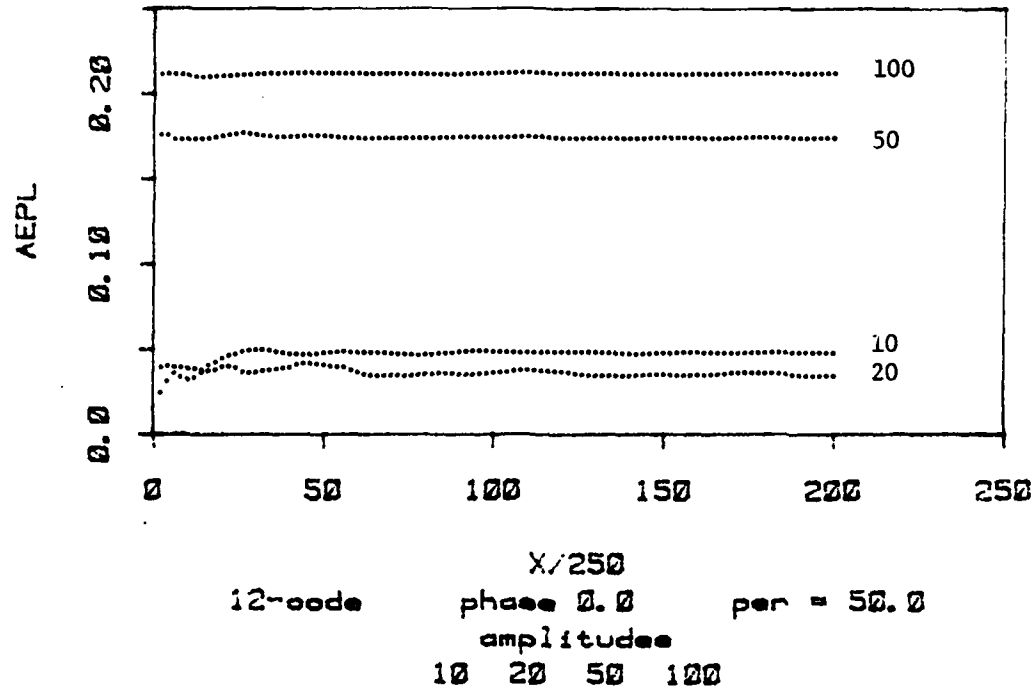


1-code phase 0.0 per = 50.0
amplitudes
10 20 50 100



Triangular Quantization

Figure A-20


AEPL
0.0
0.10
0.20
0
50
100
150
200
250
100
50
10
20
X/250
12-code phase 0.0 per = 50.0
amplitudes
10 20 50 100

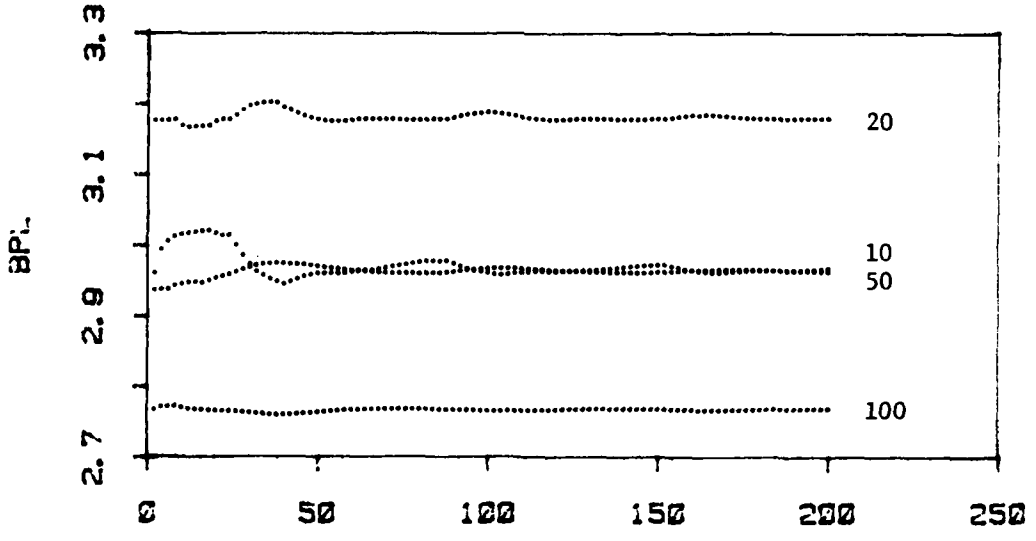
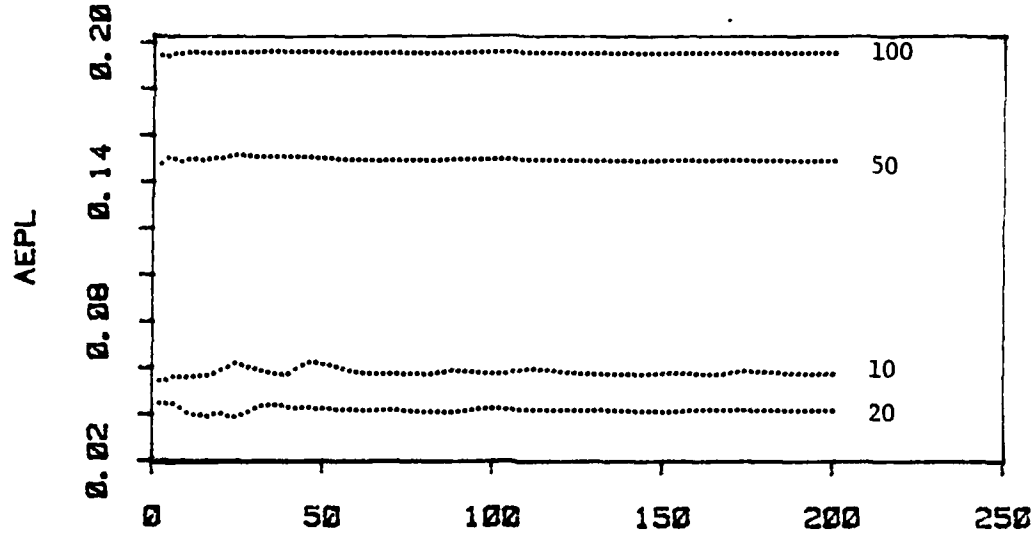

BPL
2.7
2.9
3.1
3.3
0
50
100
150
200
250
20
10
50
100
X/250
Parallel Quantization

Figure A-21

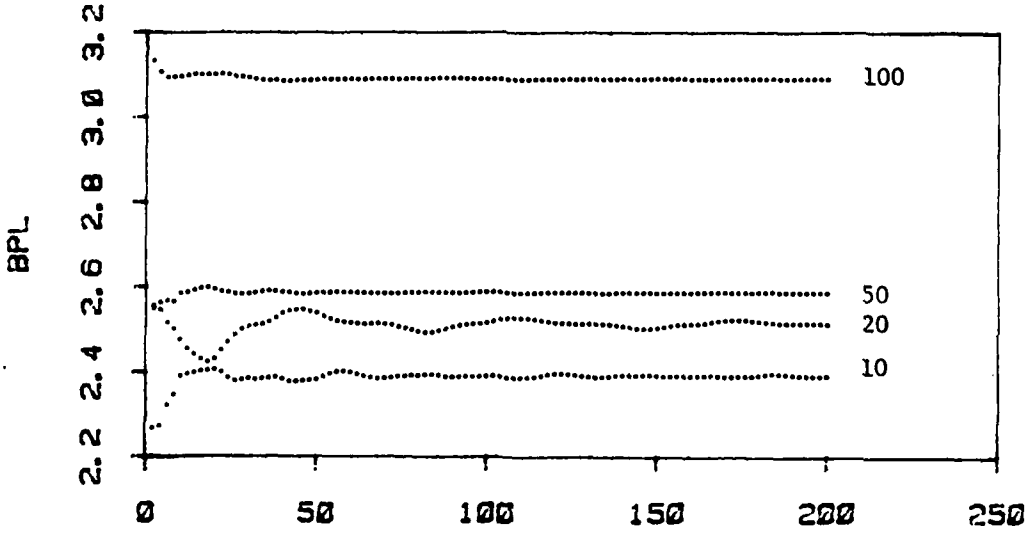


X/250

12-code phase 0.0 per = 50.0

amplitudes

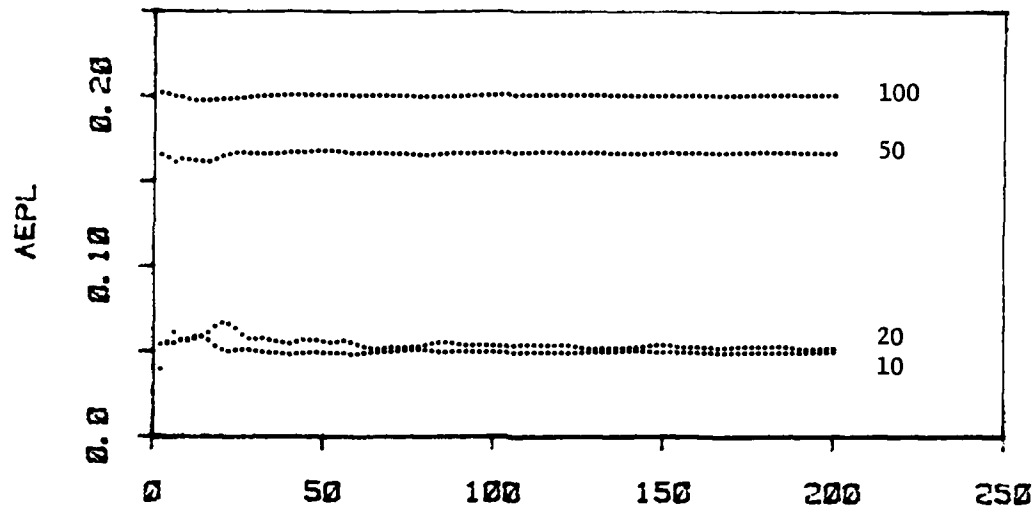
10 20 50 100



X/250

Triangular Quantization

Figure A-22

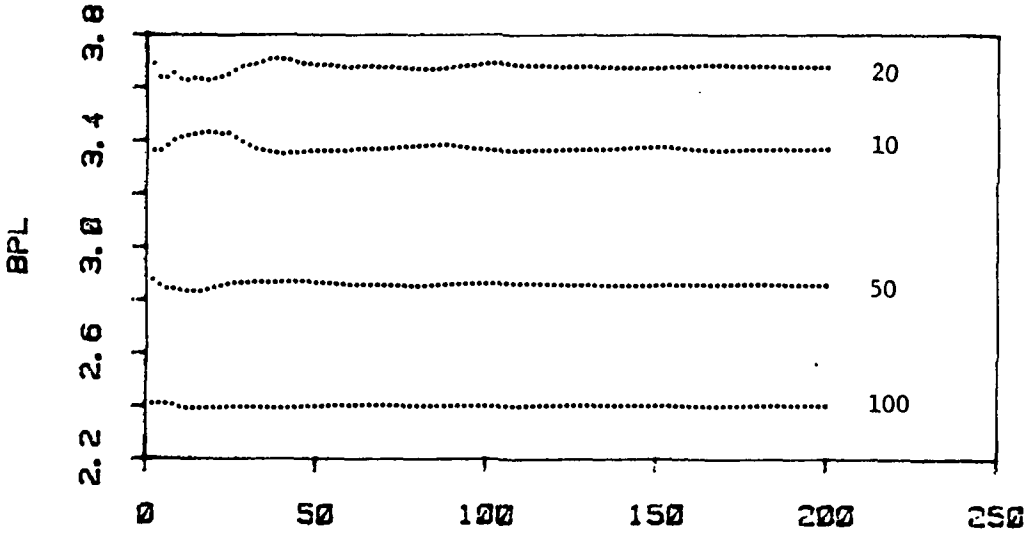


X/250

123-code phase 0.0 per = 50.0

amplitudes

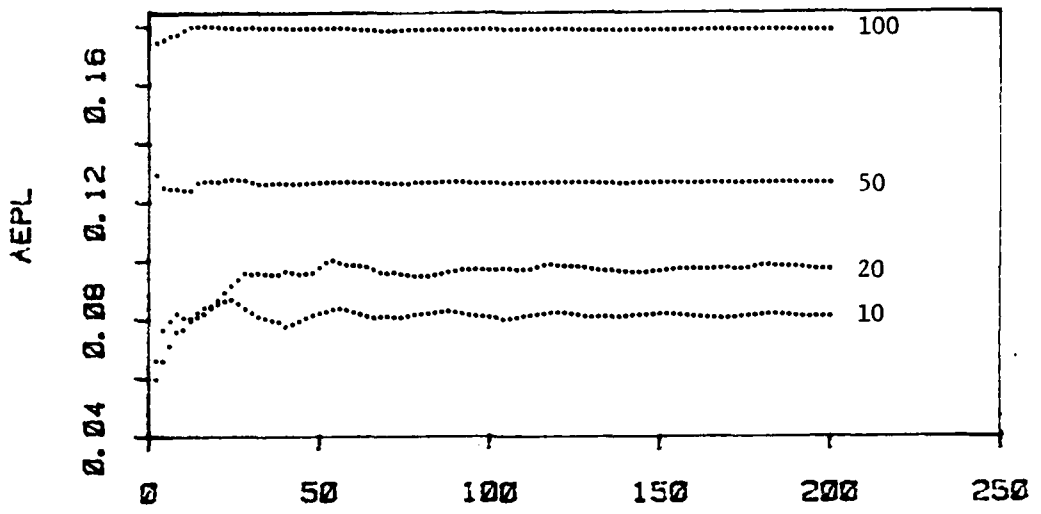
10 20 50 100



X/250

Parallel Quantization

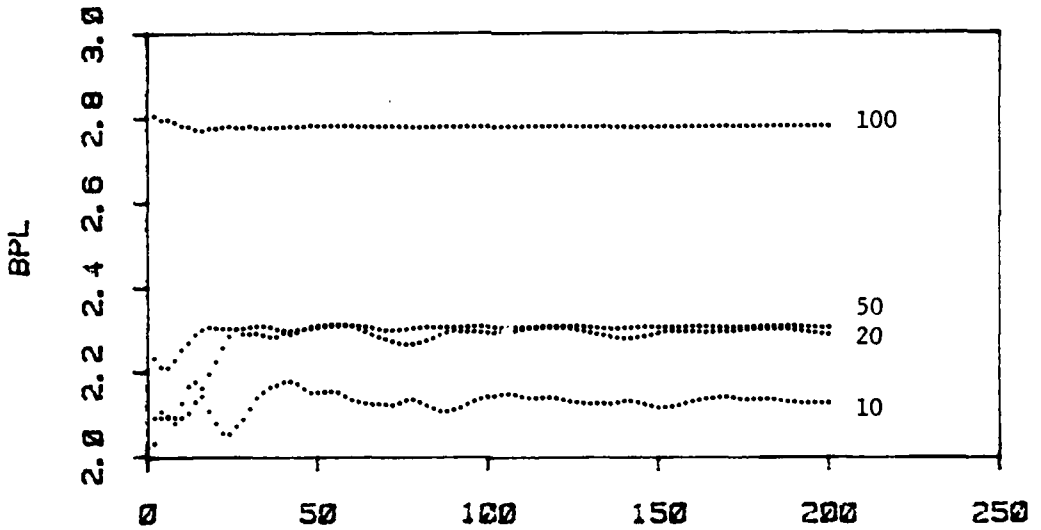
Figure A-23



123-code phase 0.0 per = 50.0

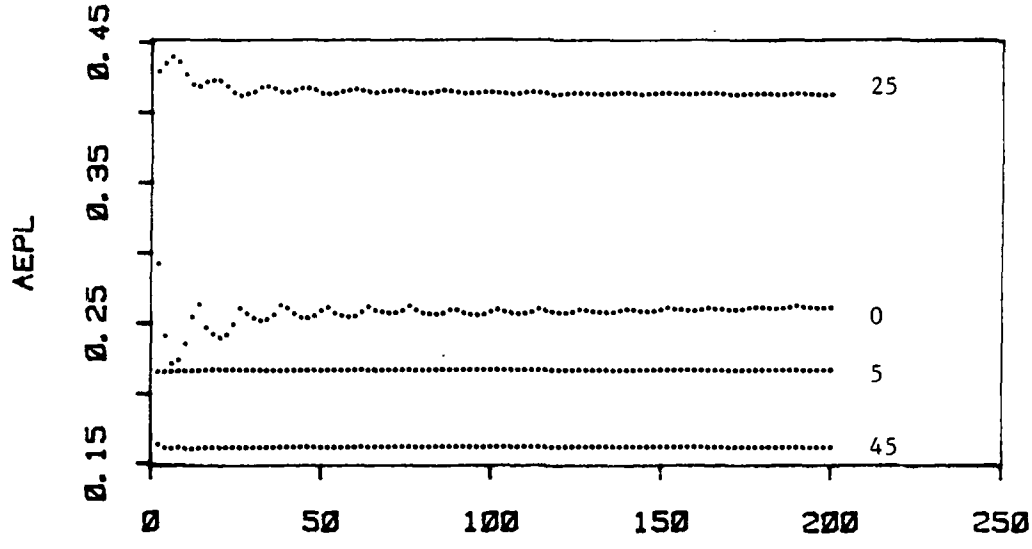
amplitudes

10 20 50 100

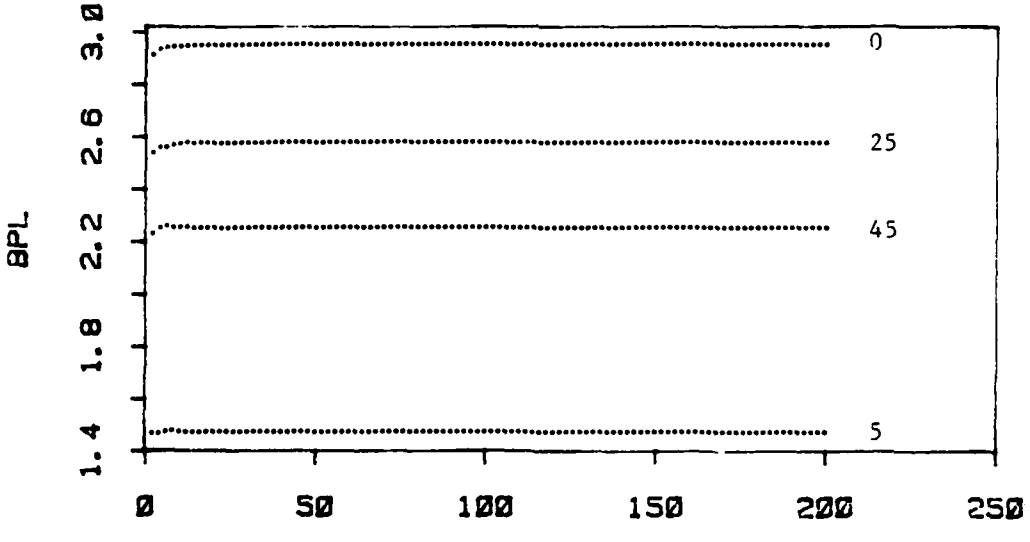


Triangular Quantization

Figure A-24

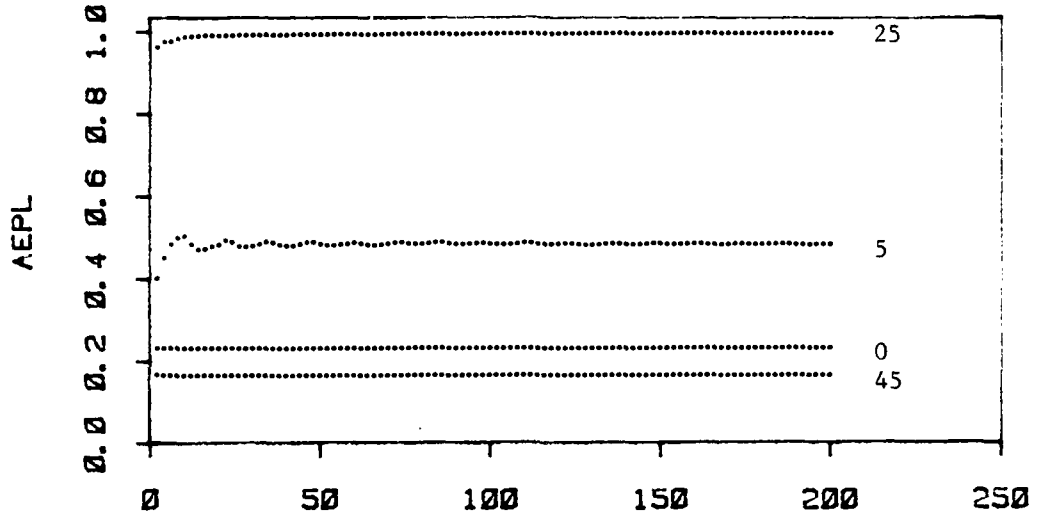


1-code phase 0.0 per = 5.0 amp = 10.0
rotations (degrees)
0 5 25 45

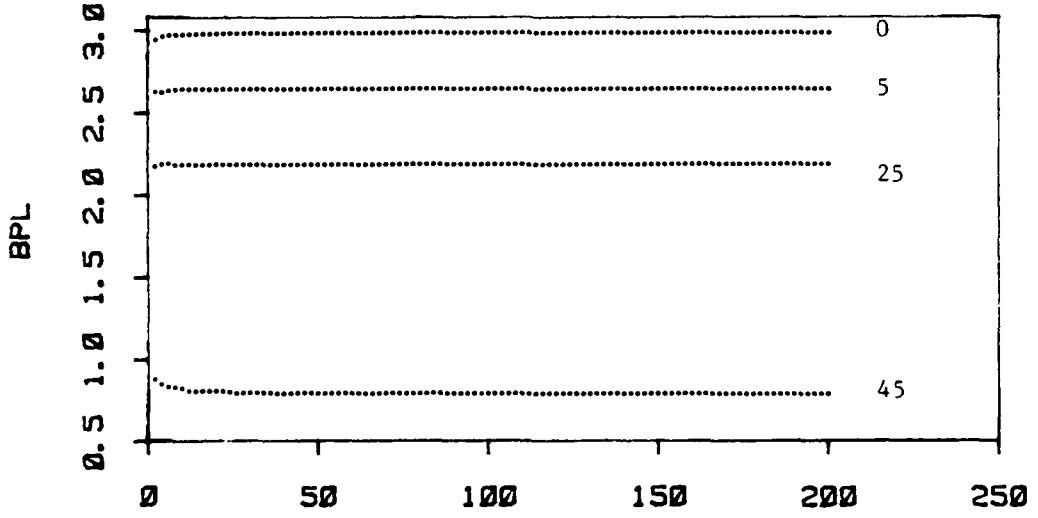


Parallel Quantization

Figure A-25

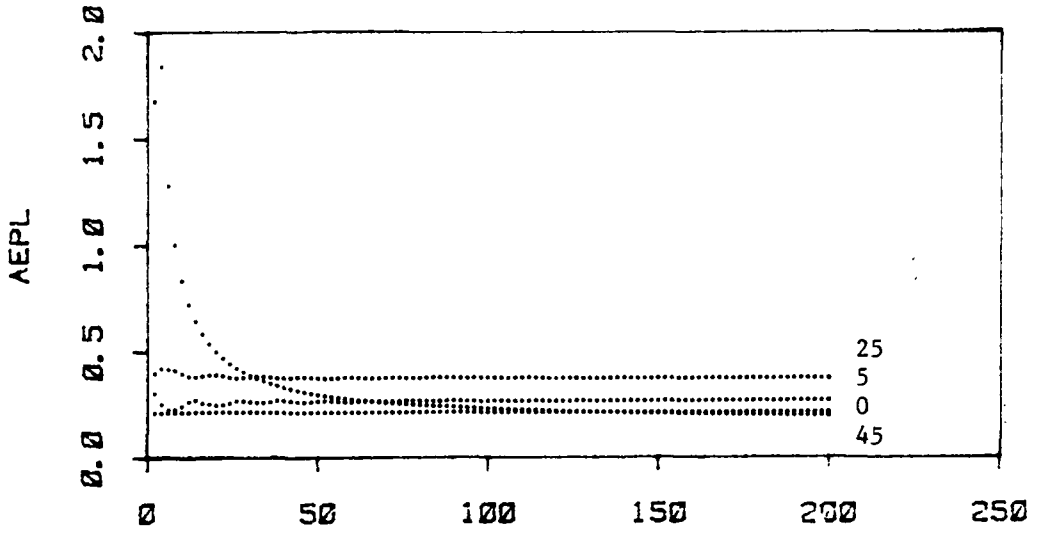


i-code phase 0.0 per = 5.0 amp = 20.0
rotations (degrees)
0 5 25 45

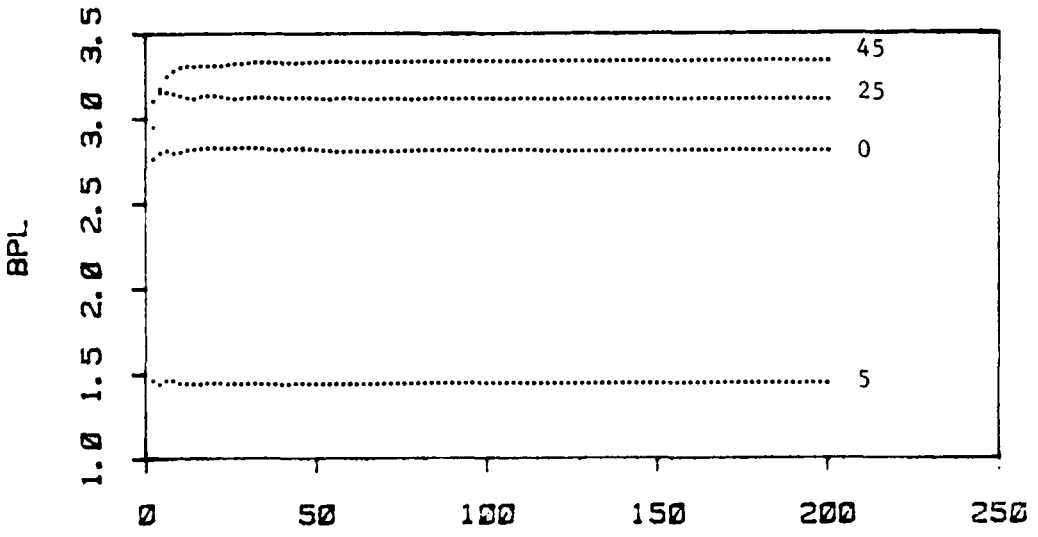


Parallel Quantization

Figure A-26



12-code phase 0.0 per = 5.0 amp = 10.0
rotations (degrees)
0 5 25 45

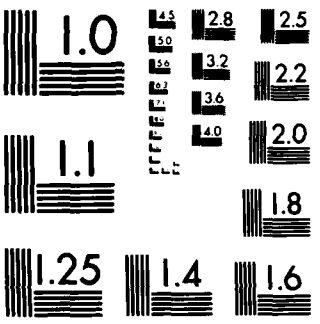


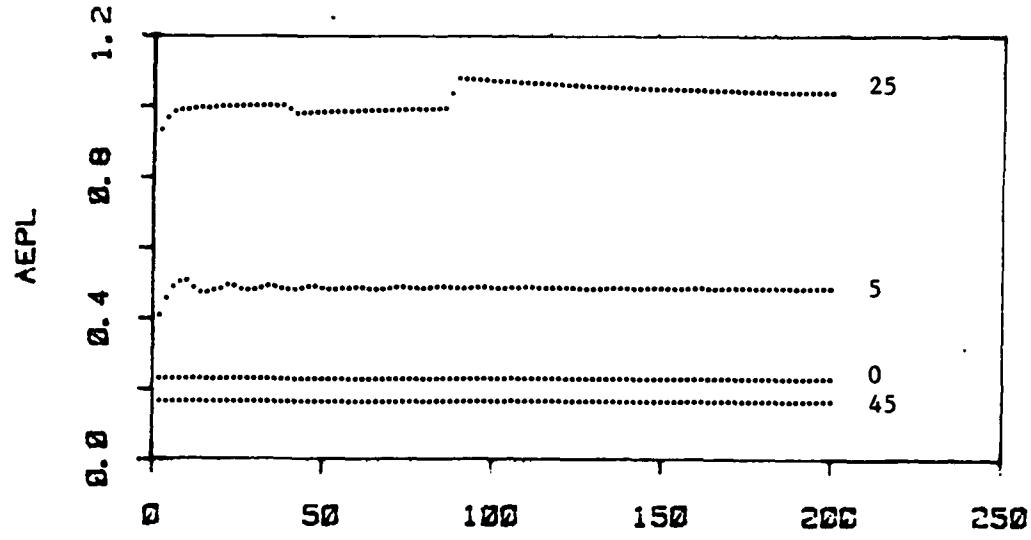
Parallel Quantization

Figure A-27

AD-A151 847 AN ANALYSIS OF GRID-BASED LINE DRAWING QUANTIZATION METHODS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING L A VALLADO DEC 84 2/2

UNCLASSIFIED AFIT/GCS/ENG/84D-31 F/G 12/1 NL



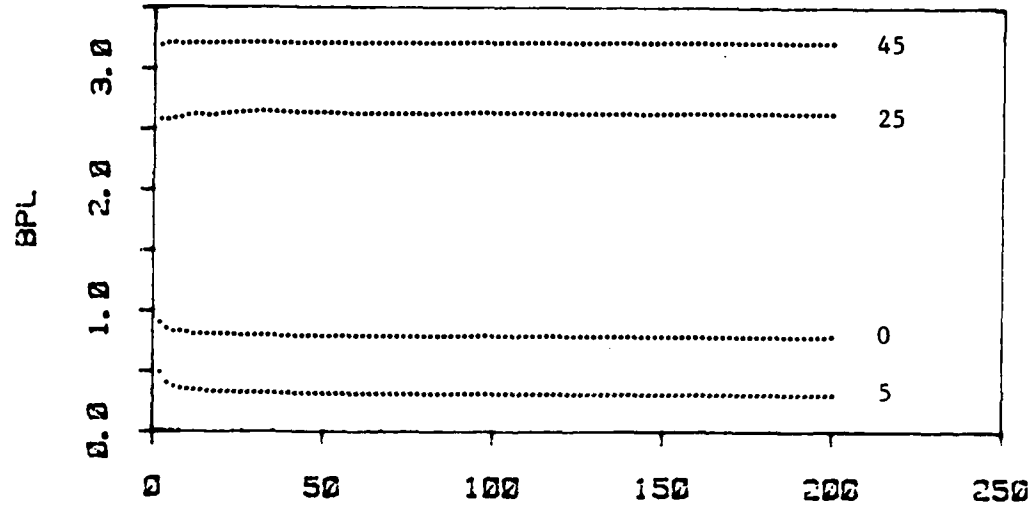


X/25

12-code phase 0.0 per = 5.0 amp = 20.0

rotations (degrees)

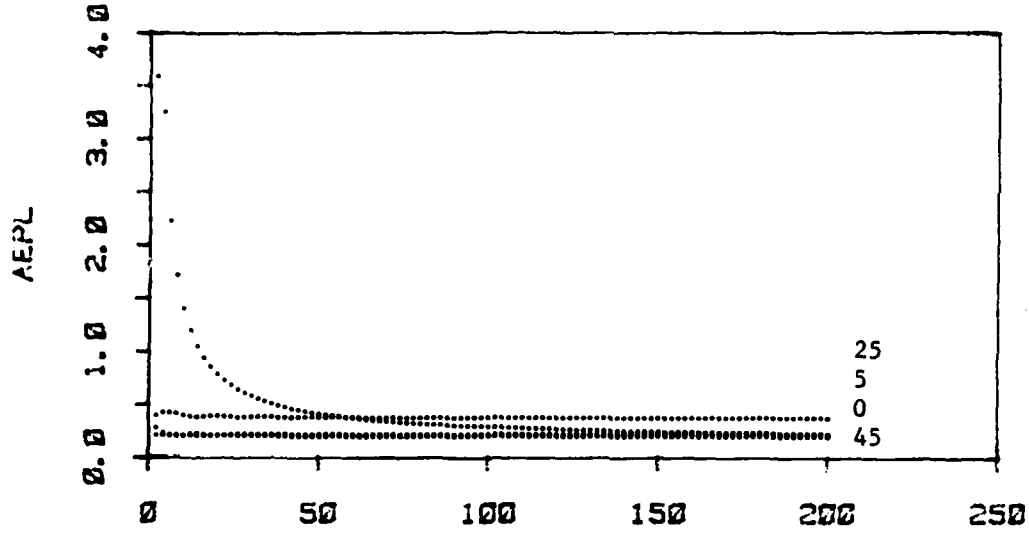
0 5 25 45



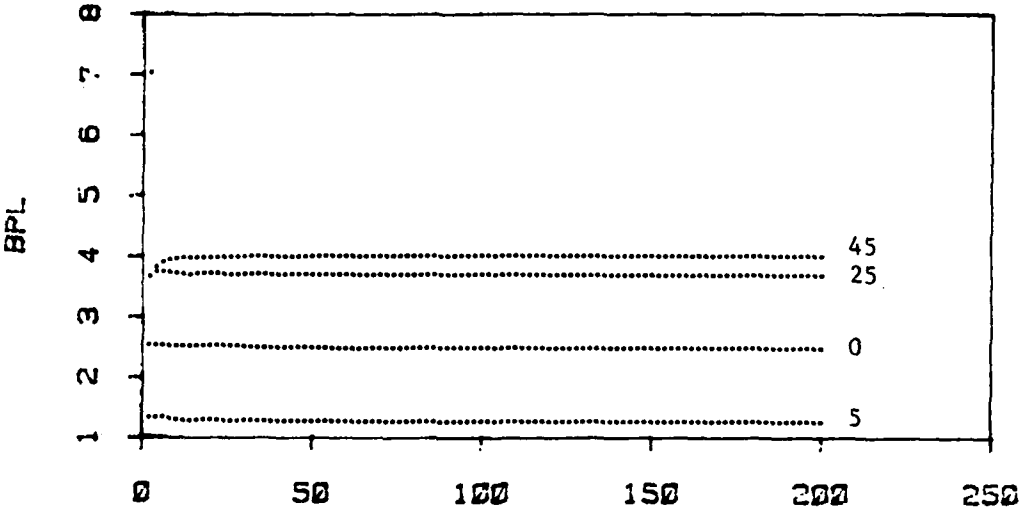
X/25

Parallel Quantization

Figure A-28

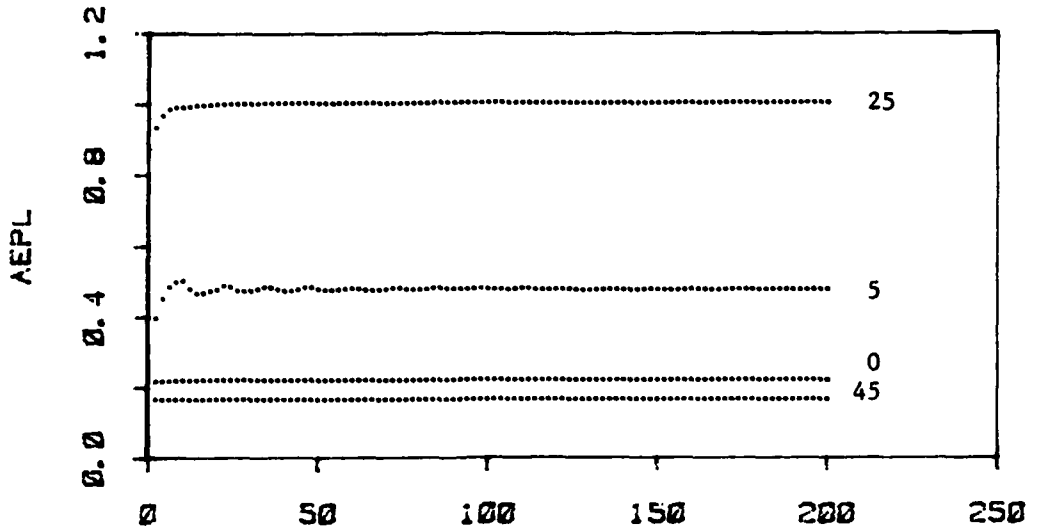


123-code phase 0.0 per = 5.0 amp = 10.0
rotations (degrees)
0 5 25 45

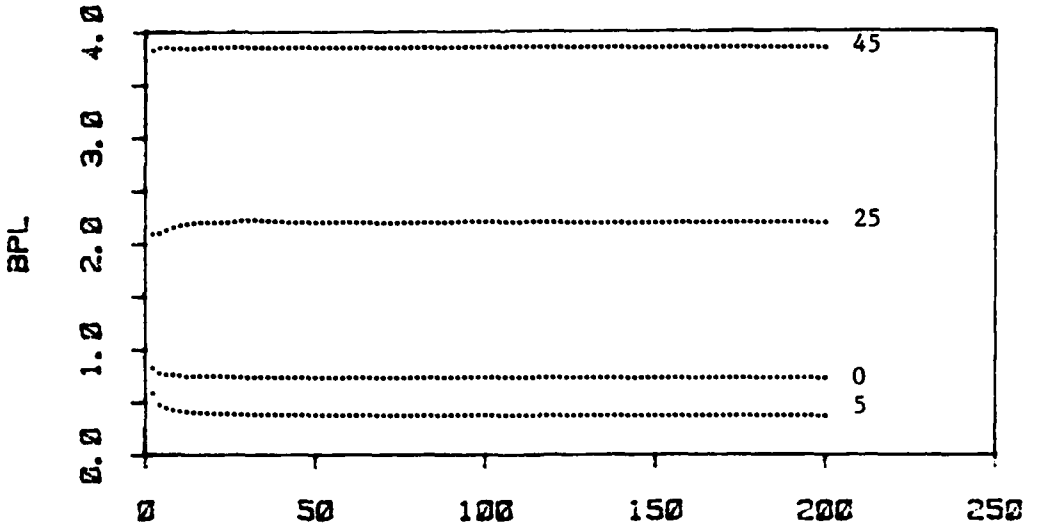


Parallel Quantization

Figure A-29

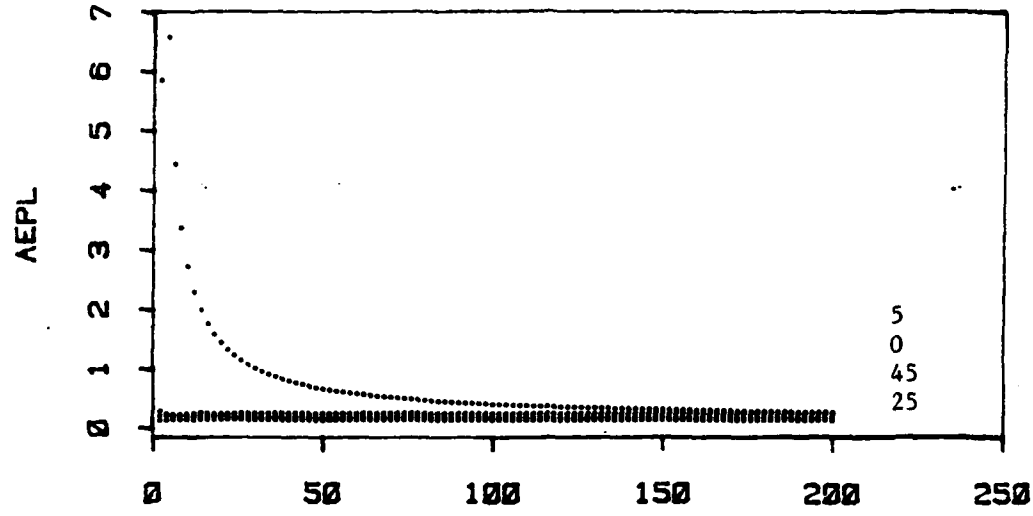


123-code phase 0.0 per = 5.0 amp = 20.0
rotations (degrees)
0 5 25 45



Parallel Quantization

Figure A-30

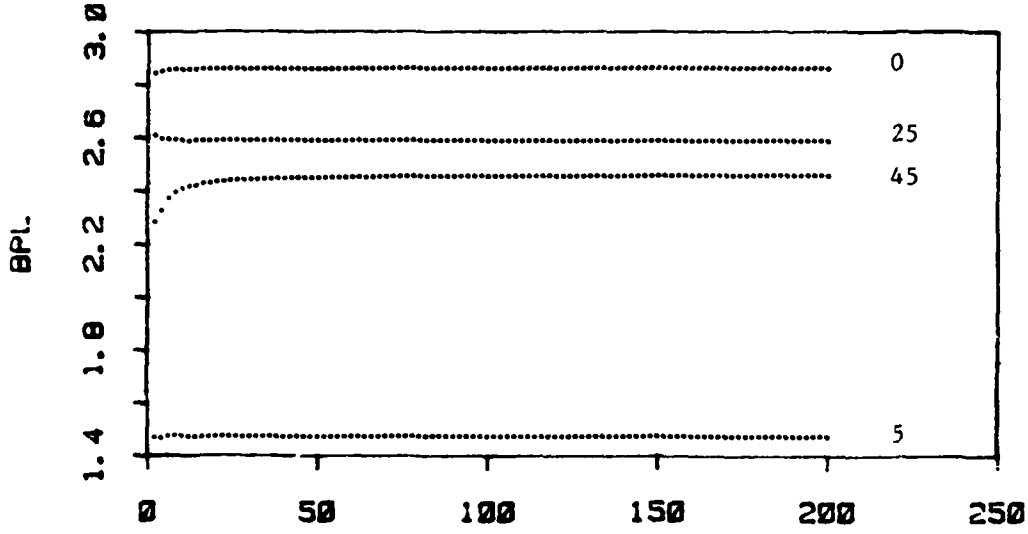


X/50

1-code phase 0.0 per = 10.0 amp = 10.0

rotations (degrees)

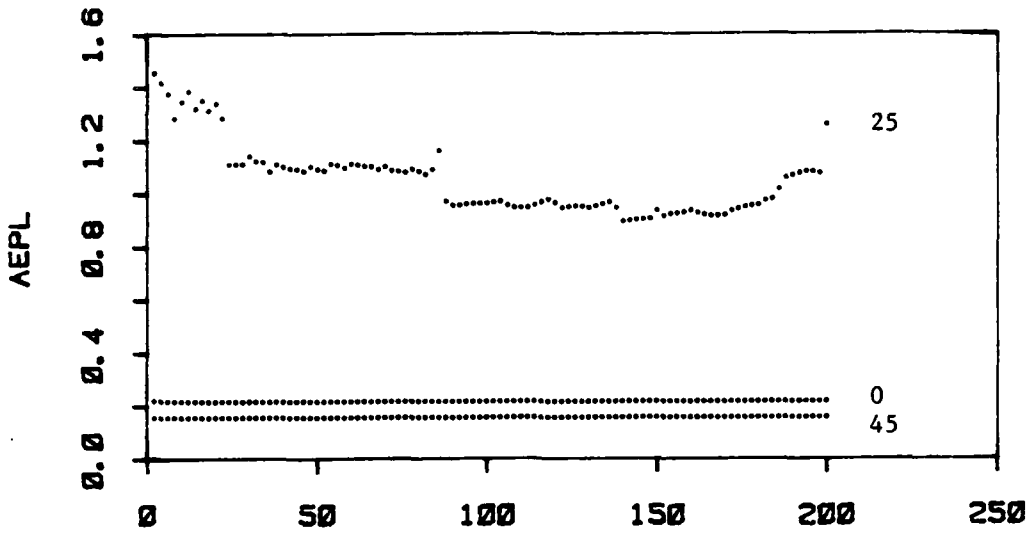
0 5 25 45



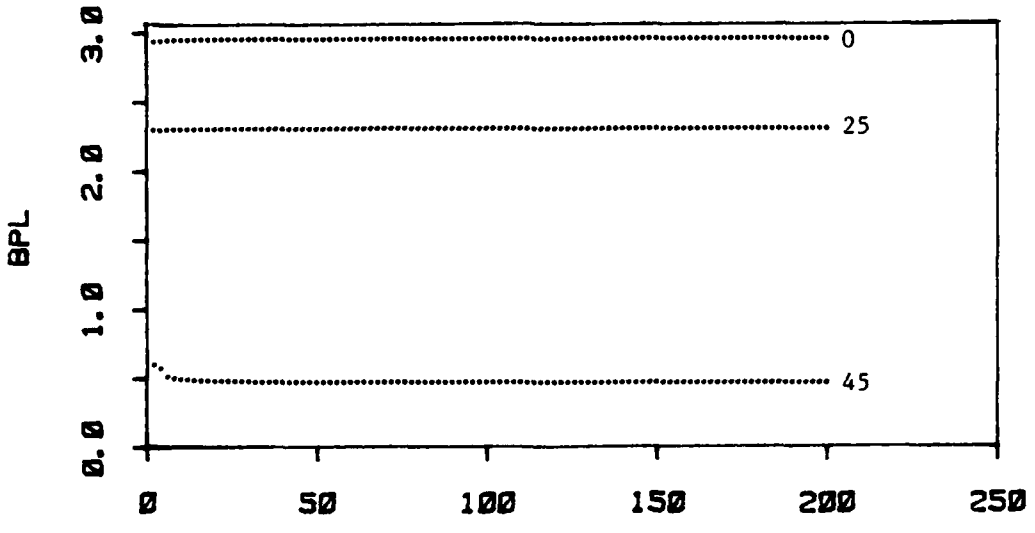
X/50

Parallel Quantization

Figure A-31

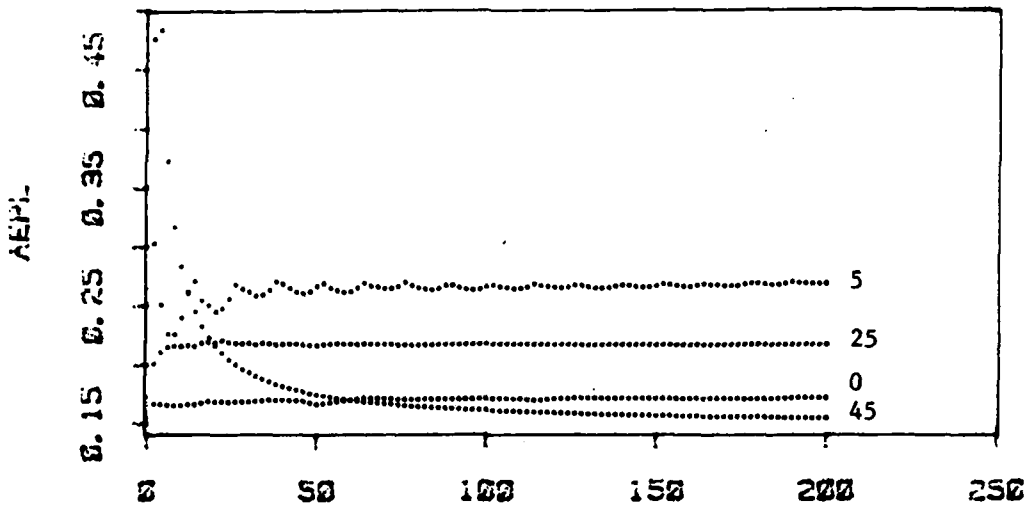


1-code phase 0.0 per = 10.0 amp = 20.0
rotations (degrees)
0 25 45



Parallel Quantization

Figure A-32

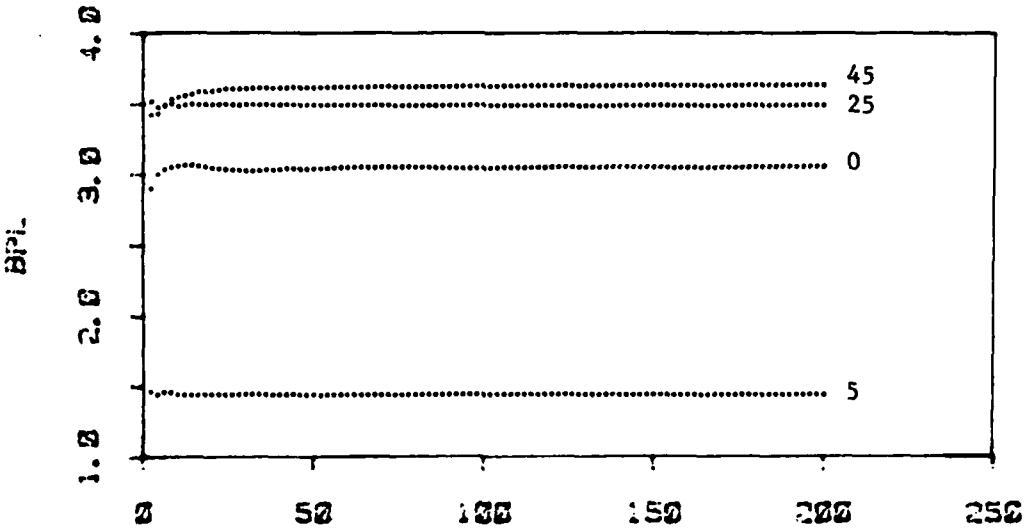


X/50

12-code phase 0.0 per = 12.2 amp = 12.0

rotations

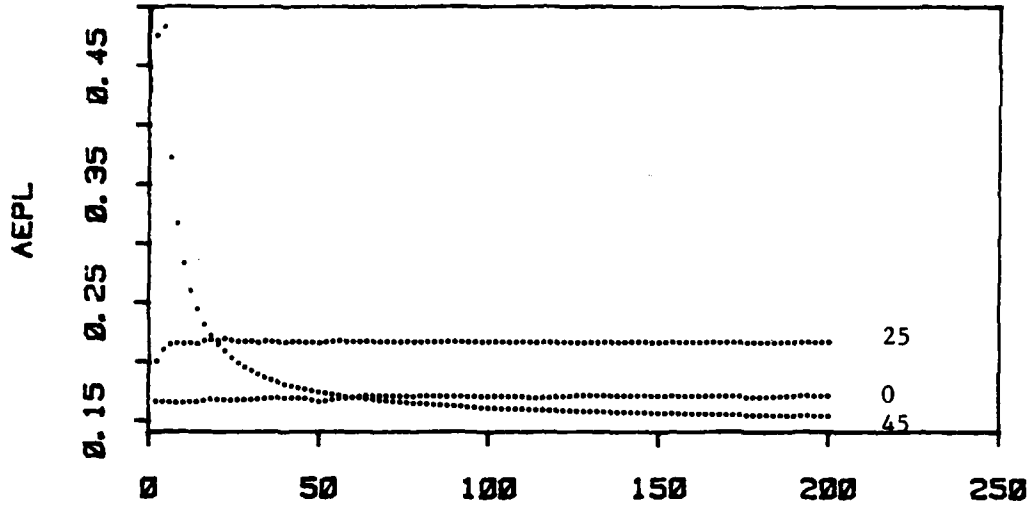
0 5 25 45



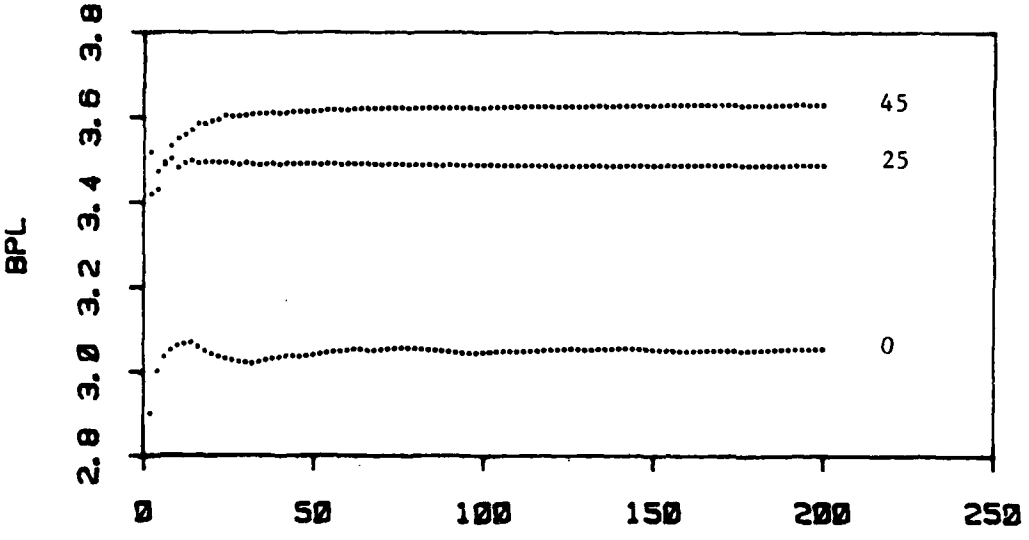
X/50

Parallel Quantization

Figure A-33

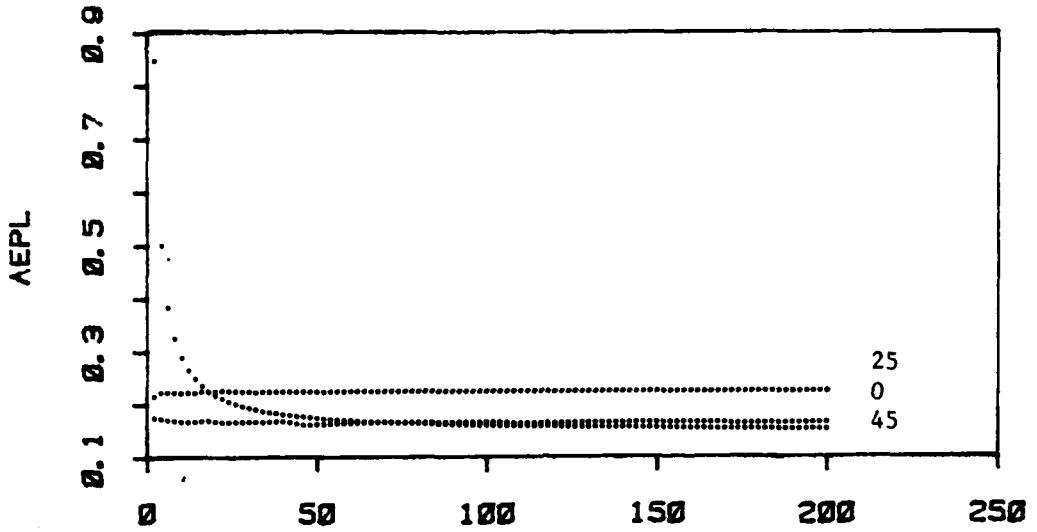


12-code phase 0.0 per = 10.0 amp = 20.0
rotations (degrees)
0 25 45

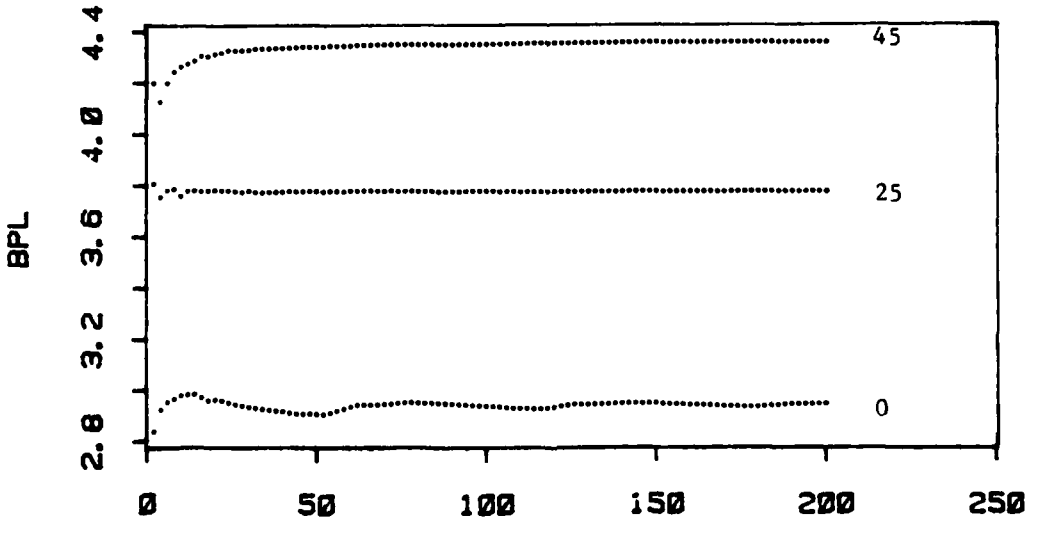


Parallel Quantization

Figure A-34

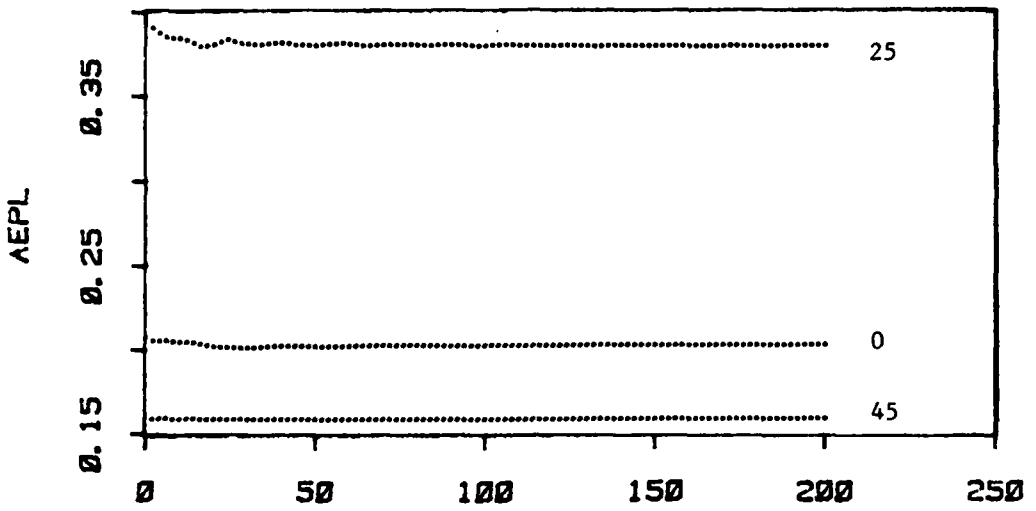


123-code phase 0.0 per = 10.0 amp = 10.0
rotations (degrees)
0 25 45

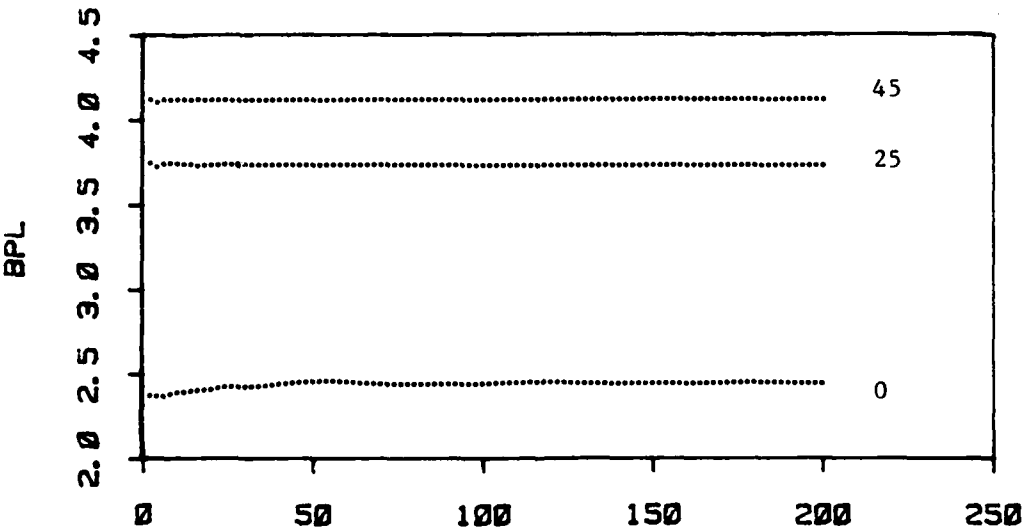


Parallel Quantization

Figure A-35

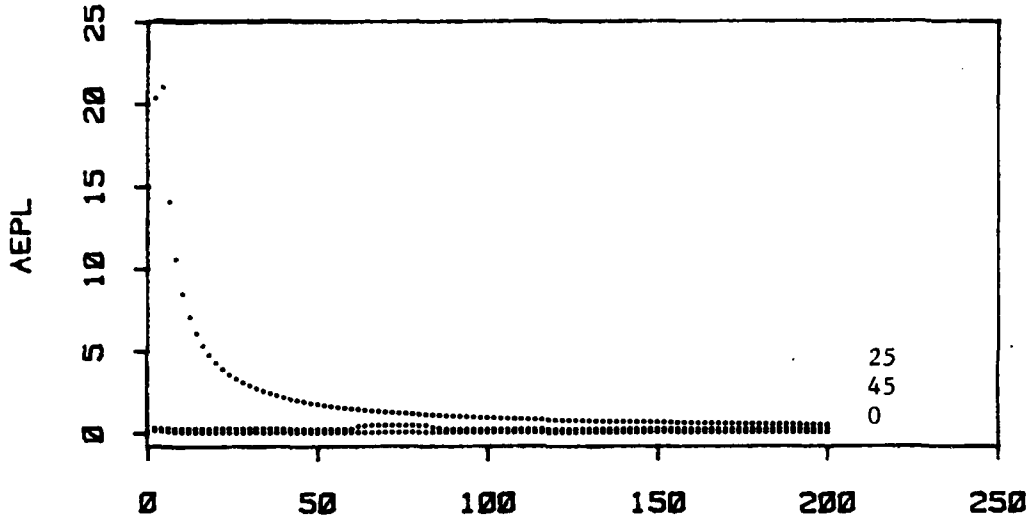


123-code phase 0.0 per = 10.0 amp = 20.0
rotations (degrees)
0 25 45

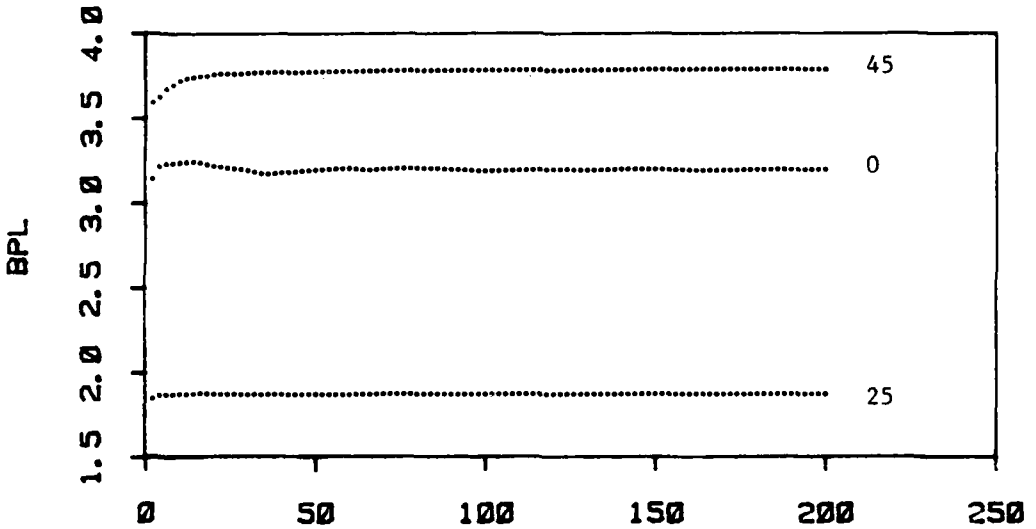


Parallel Quantization

Figure A-36

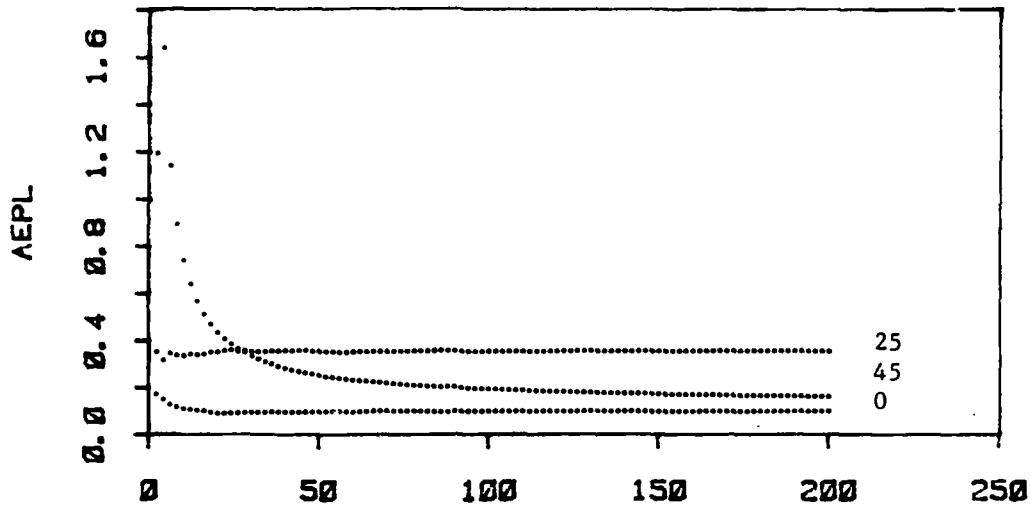


12-code phase 0.0 per = 20.0 amp = 10.0
rotations (degrees)
0 25 45

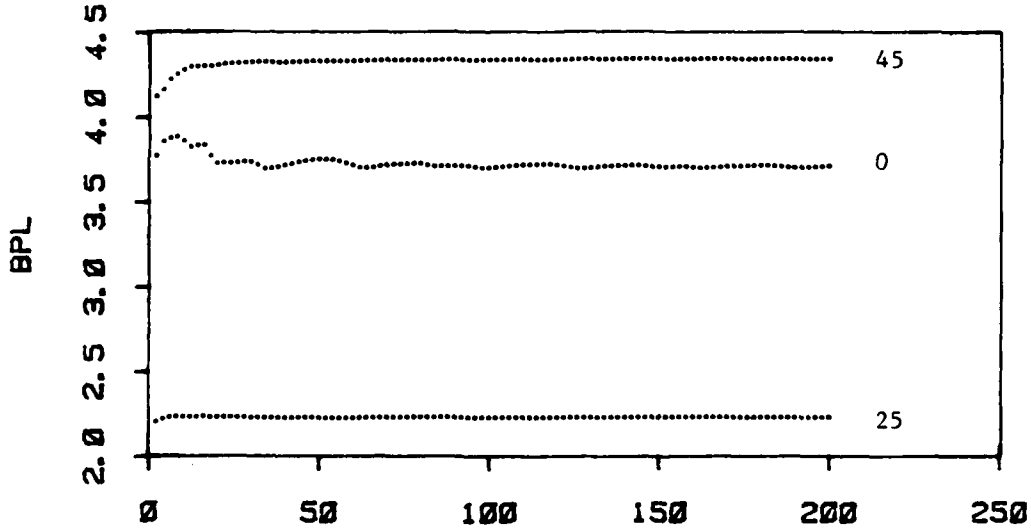


Parallel Quantization

Figure A-37



123-code phase 0.0 per = 20.0 amp = 10.0
rotations (degrees)
0 25 45



Parallel Quantization

Figure A-38

TABLE A-1

Asymptotic Values of AEPL and BPL

Code	Per	Amp	Parallel AEPL	Parallel BPL	Triangular AEPL	Triangular BPL
1	5	10	.2169148	2.956647	All triangular is	
		20	.2324887	2.985966	the same for the	
		50	.2417930	2.996777	(1)-code as the	
		100	.2448190	2.997401	parallel.	
	10	10	.1912305	2.865060		
		20	.2190135	2.957843		
		50	.2368649	2.991154		
		100	.2431569	2.996960		
	20	10	.1417753	2.726084		
		20	.1916683	2.879882		
		50	.2250814	2.971457		
		100	.2373460	2.991317		
	50	10	.03315402	2.431296		
		20	.1124414	2.654535		
		50	.1890147	2.899552		
		100	.2191669	2.964922		
12	5	10	.2117598	2.811842	.1949870	3.118601
		20	.2293675	2.985966	.2198348	3.392349
		50	.2404755	2.545724	.2358918	3.596373
		100	.2441583	2.520583	.2418611	3.672662
	10	10	.1715845	3.055777	.1509113	2.677445
		20	.2115526	2.792732	.1942909	3.111813
		50	.2344184	2.617398	.2260701	3.455598
		100	.2417099	2.556388	.2373043	3.593914
	20	10	.1038196	3.198367	.0776187	2.788676
		20	.1780185	2.966964	.1603534	2.604717
		50	.2195363	2.734720	.2045522	3.193342
		100	.2347226	2.621659	.2261768	3.452637
	50	10	.0479690	2.967515	.0576142	2.390972
		20	.0345809	3.178946	.0418754	2.512943
		50	.1740581	2.962589	.1492145	2.587076
		100	.2120605	2.768713	.1956103	3.090855

Code	Per	Amp	Parallel AEPL	Parallel BPL	Triangular AEPL	Triangular BPL
123	5	10	.1976350	2.482167	.1750795	2.846140
		20	.2209712	2.199789	.2086007	3.375976
		50	.2362195	2.058378	.2306464	3.835452
		100	.2418210	2.021018	.2390865	3.997254
	10	10	.1675558	2.945834	.1177411	2.418652
		20	.2026676	2.448016	.1807891	2.837347
		50	.2269409	2.157740	.2169555	3.527935
		100	.2375966	2.073932	.2320895	3.833226
	20	10	.1007630	3.711815	.0675241	2.560038
		20	.1684065	2.869498	.1399304	2.281905
		50	.2069011	2.338043	.1887736	2.999730
		100	.2162931	3.522135	.2162931	3.522135
	50	10	.0497579	3.369408	.0808088	2.124164
		20	.0518447	3.677714	.0969815	2.286100
		50	.1663073	2.855613	.1263413	2.303243
		100	.2002311	2.401456	.1784513	2.779536

TABLE A-2

Asymptotic Values of AEPL and BPL - Rotated

Code	Per	Rot	Amp 10 AEPL	Amp 10 BPL	Amp 20 AEPL	Amp 20 BPL
1	5	0	.2169148	2.956647	.2324887	2.985966
		5	.2164732	1.474501	.3481222	2.900685
		25	.4127414	2.581256	.9958125	2.643763
		45	.1620588	2.256050	.1661394	2.186905
12		0	.2117598	2.811842	.2293675	2.628679
		5	.2685257	1.444509	.3493312	2.597236
		25	.3714814	3.113185	.6240516	3.085982
		45	.1617397	3.345992	.1659109	3.214331
123		0	.1976350	2.482167	.2209712	2.199789
		5	.2193143	1.263055	.3372156	2.184388
		25	.3712354	3.68869	.6215464	3.702656
		45	.1617397	4.015191	.1659109	3.857197
1	10	0	.1912305	2.865060	.2190135	2.957843
		5	.2614732	1.474501		
		25	.1421677	2.590801	.4166651	2.650670
		45	.1506802	2.463147	.1592000	2.306509
12		0	.1715845	3.055777	.2115526	2.792732
		5	.2685257	1.444509		
		25	.2170967	3.487819	.3788468	3.168386
		45	.1544718	3.630939	.1589318	3.435961
123		0	.1675558	2.945834	.2026676	2.448016
		5				
		25	.2264479	3.773633	.3798732	3.733440
		45	.1548153	4.357270	.1589318	4.123153
1	20	0	.1417753	2.726084	.1916683	2.879882
		5				
		25			.2219781	3.779683
		45	.0992579	2.741957	.1416104	2.505992
12		0	.1038196	3.198367	.1780185	2.966964
		5				
		25	.3183834	1.875271	.2072149	3.511271
		45	.1416206	3.791665	.1436359	3.682828
123		0	.1007630	3.711815	.1684065	2.863498
		5				
		25	.3548583	2.233014	.1357985	3.779683
		45	.1628572	4.344398	.1491131	4.403397

Vita

Laura Ann (Hale) Vallado was born on 20 August 1959 at Vandenberg AFB, California. She graduated from Franklin Community High School, Franklin, Indiana, in May of 1977. She attended the United States Naval Academy and graduated from Franklin College in May of 1980 with a B.A. in mathematics. She attended Officer Training School and was commissioned a Second Lieutenant in the United States Air Force on 24 October 1980. She attended the computer programmer's technical training at Keesler AFB, Mississippi, and was then assigned to the 7102 Computer Services Squadron at the Headquarters United States Air Forces in Europe in February 1981. She then applied and was accepted at the Air Force Institute of Technology in May of 1983. She married a classmate, David Anthony Vallado on 18 March 1984.

Permenent Address: 3 Glencove Rd
Morris Plains, New Jersey

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/84D-31	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Laura A. Vallado, B.A., Capt, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1984 December 15. PAGE COUNT 104
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Parallel Quantization Scheme, Triangular Quan. Scheme, Line Drawing Quantization, Generalized Chain Codes
09	02		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: An Analysis of Grid-Based Line Drawing Quantization Methods
Thesis Chairman: Ken Castor, Major, USAF

Approved for public release: IAW AFR 190-17.
LYNN E. [illegible] 21 Feb 85
Dean for Research and Professional Development
Air Force Institute of Technology (AIC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Ken Castor, Major, USAF	22b. TELEPHONE NUMBER (Include Area Code) 513-255-3576 22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

This document implements the parallel quantization scheme for quantizing line drawings. It also discusses the theory behind an algorithm for a rotated drawing quantization algorithm. The algorithm was also implemented during this thesis. Both of these algorithms were analyzed and compared. The parallel quantization scheme was compared with the previously implemented triangular quantization scheme. The rotated drawing algorithm was implemented using the parallel quantization scheme and compared using different rotations.

The results of these computer programs were analyzed using two criterion: compactness and accuracy. Then recommendations were made using these results.

END

FILMED

4-85

DTIC